BIBLIOTECA BASICA INFORMATICA

DISEÑO DE JUEGOS 17



INGELEK

BIBLIOTECA BASICA INFORMATICA

DISEÑO DE JUEGOS 17

INGELEK

Director editor:
Antonio M. Ferrer Abelló.

Director de producción:
Vicente Robles.

Coordinador y supervisión técnica:
Enrique Monsalve.

Colaboradores:
Angel Segado
Casimiro Zaragoza
Fernando Ruíz
Francisco Ruíz
Jesús Pedraza
Juanjo Alba Ríos
Margarita Caffaratto
María Angeles Gálvez
Marina Caffaratto
Masé González Balandín
Patricia Mordini

Diseño:
Bravo/Lofish.

Dibujos:
José Ochoa.

ISBN del tomo: 84-85831-51-9
ISBN de la obra: 84-85831-31-4
Fotocomposición: Pérez Díaz, S. A.
Imprime: Héroes, S. A.
Depósito Legal: M-4866-1986
Precio en Canarias, Ceuta y Melilla: 380 pts.

INDICE

PROLOGO

En este libro pretendemos que todos acaben dominando el "arte" del diseño y programación de juegos en su ordenador. Si no saben lo que es un juego de aventuras podrán reparar esa grave laguna intelectual de su formación leyendo simplemente el capítulo 1 de este libro. Si ya están informados (o son expertos aventureros) podrán saltarse el capítulo o bien leerlo por si pueden enriquecer su bagaje cultural con nuevas informaciones. De todas formas, la elección es suya: los lectores son ustedes.

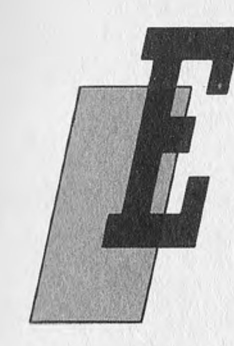
También nos proponemos, ya de paso, insistir un poco sobre el BASIC y el "estilo" de programación. "Ludendo docere" (enseñar jugando) era la expresión que se podría haber usado muchos siglos: esto es lo que intentamos, dada nuestra aversión hacia cualquier forma de enseñanza inútilmente aburrida o "profesoral", sin ánimo de querer ofender (todo lo contrario, con todos los respectos) a los profesores verdaderos, aquellos que saben activar el mecanismo más importante necesario para aprender: el interés.

¡Que disfruten!

CAPITULO I

LA AVENTURA DE LOS JUEGOS DE AVENTURAS

Los orígenes



En una fecha que no hemos conseguido saber con exactitud, pero que de todas formas es anterior a la existencia de los ordenadores personales (más o menos en los primeros años de la década de los setenta), Crowther y Woods inventaron una nueva modalidad de juego, en la cual un ordenador describe un ambiente, o una situación, con frases como ésta (traducimos libremente):

Estás en una espléndida habitación de unos diez metros de altura. Los muros son ríos congelados de color anaranjado. Un abrupto cañón y un paso fácil salen de los lados este y oeste de la habitación.
Hay un alegre pajarillo piando.

El jugador, que se identifica como el personaje que está investigando el misterioso lugar, puede comunicarse con el ordenador por medio de frases sencillas escritas en el teclado, por ejemplo:

Captura el pajarillo.

Obteniendo como respuesta:

El pajarillo no tenía miedo cuando has entrado, pero al acercarte se inquieta y no consigues capturarlo.

De esta respuesta, el jugador obtiene algunas informaciones: por ejemplo, que el pajarillo tiene miedo de él, o bien, tiene miedo de algún objeto que él posee. Efectivamente, también es posible coger o dejar diferentes objetos (la finalidad del juego es encontrar todos los tesoros y salir intacto del lugar) y trasladarse de un lugar a otro.

El ordenador se comporta como si "entendiese" las frases del jugador, dando la impresión de que realmente "vive" las situaciones descritas.

Crowther y Woods llamaron a su juego simplemente "Adventure", es decir, "Aventura". Un nombre muy apropiado que desde entonces sirve para referirse a todos los juegos del mismo tipo.

Desde los "minis" a los ordenadores personales

Aunque el programa se escribió en un minicomputador, o sea, una máquina relativamente pequeña para la época (concretamente un DEC PDP-10), el Adventure original ciertamente no estaba destinado al gran público: en aquellos tiempos (que parecen tan lejanos) los ordenadores eran máquinas costosísimas, sólo accesibles a unos pocos especialistas y patrimonio de grandes industrias, universidades y centros de investigación.

Aún así, el juego fue un gran éxito entre los pocos afortunados que pudieron jugar con él. También fue causa de muchos problemas en los centros de trabajo, porque los directores veían cómo los programadores se distraían con problemas que tenían muy poco que ver con su trabajo. Según parece, todas las prohibiciones fueron inútiles: lo único posible era esperar a que el juego acabara, entonces el trabajo podía reanudarse.

Con la introducción de los ordenadores personales varios programadores se lanzaron a la tarea de crear juegos de aventuras que funcionaran en las nuevas máquinas. El asunto no era nada fácil: basta pensar que el Adventure original (escrito en lenguaje FORTRAN) ocupaba casi 300 kbytes, mientras que en los ordenadores personales había disponibles sólo 16 kbytes, más o menos.

Uno de los precursores de los juegos de aventuras en los ordenadores personales fue Scott Adams, que ya en 1978 escribió (en BASIC) un intérprete que le permitió crear pequeñas pero bien organizadas aventuras (explicaremos lo que es un intérprete dentro de algunos capítulos).

Más adelante, la evolución de los pequeños ordenadores permitió escribir aventuras cada vez más complejas, incluso versiones del Adventure original (para el Apple II, por ejemplo, existen dos: Apple Adventure y Microsoft Adventure, prácticamente idénticas).

¿Texto o gráficos?

El diálogo escrito no es el único medio posible de comunicación entre jugador y ordenador: un ambiente o una situación también se pueden dibujar y no sólo describir. Por lo tanto, existe también un gran número (y en constante aumento) de aventuras gráficas; existe una especie de contencioso entre los partidarios de los dos tipos de juegos. Personalmente, nosotros creemos que las dos formas tienen sus cosas buenas: si bien una imagen en color es más atrayente que un texto escrito, también es verdad que el segundo estimula más la fantasía; ningún dibujo estará nunca a la altura de una escena imaginada en base a una descripción sugerente. Es la misma diferencia que existe entre un buen libro y una buena película.

Hacemos notar que hemos dicho "un buen libro" y "una buena película". Entendemos que ambos tipos son válidos si están bien hechos, pero hay una verdadera avalancha de aventuras decididamente malas (de ambos géneros). Retomaremos este tema más adelante para intentar explicar los factores que contribuyen a la calidad de una aventura.

En este libro les enseñaremos a construir aventuras de tipo texto, pero los mismos principios (como veremos) son aplicables perfectamente a la construcción de aventuras gráficas.

CAPITULO II

LA NAVE ESPACIAL CONDENADA

Introduciéndonos en el programa

No es nuestra intención aburrirles con confusos planteamientos teóricos. En cambio, les proponemos enseguida una aventura. En los próximos capítulos les mostraremos cómo ha nacido este juego, paso a paso. Aprenderán así a diseñar "a medida" sus propias aventuras, aprovechando, si lo desean, una parte (la más compleja) del programa que les proponemos, y añadiéndole los frutos de su fantasía. Finalmente, podrán ver cómo funciona el programa y cómo modificarlo según sus exigencias particulares. Habrá también una cierta dosis de teoría, pero diluida y (esperamos) digerible con facilidad. Este libro requiere un conocimiento por lo menos superficial del BASIC. Si no lo tienen, o si quieren mejorarlo, pueden acudir a los libros 5, 6, 7 y 9 de nuestra colección.

El listado de la aventura espacial aparece al final de este capítulo. Si no tienen ganas de teclear las más de 350 líneas que constituyen el programa, pueden encontrarlo ya preparado en "JACKSON SOFT AVVENTURE 1", también ligeramente modificado.

El listado que publicamos está en versión Apple II, pero para muchos otros ordenadores incluimos las variaciones necesarias. Una única excepción importante: la versión para el Spectrum, que utiliza un BASIC no estándar y bastante limitado. Si tienen un Spectrum, sólo hace falta que se provean del citado número de JACKSON SOFT AVENTURAS, cuya casete contiene una versión del juego oportunamente modificado.

Adaptación

- Apple II (todas las versiones): ninguna variación;
- Commodore 64 o VIC-20 expandido (por lo menos 16K): quitar el punto de interrogación en las INPUT de las líneas 800 y 1330 y sustituir las líneas 1990-2090 por las que aparecen en la figura 1.
- MSX, IBM, Olivetti y Macintosh con microsoft BASIC: quitar el punto de interrogación en las INPUT de las líneas 800 y 1330, sustituir la línea 1130 por:

  1130 A$ = INPUT$ (1)

  y sustituir las líneas 1990-2090 por las que aparecen en la figura 2.

```
1990 REM 5:SAVE
2000 OPEN 1,1,1,F$
2010 FOR I=1 TO FO:PRINT#1,LO%(I):NEXT I
2020 PRINT#1,LU:PRINT#1,T1:PRINT#1,V1:PRINT#1,V2
2030 CLOSE 1:RETURN
2050 REM 6:LOAD
2060 OPEN 1,1,0,F$
2070 FOR I=1 TO FO:INPUT#1,LO%(I):NEXT I
2080 INPUT#1,LU,T1,V1,V2
1090 CLOSE 1:RETURN
```

Figura 1.—Subrutinas de "Save" y "Load" para el Commodore 64 y el Vic 20 (ampliado).

```
1990 REM 5:SAVE
2000 OPEN F$ FOR OUTPUT AS #1
2010 FOR I=1 TO FO:PRINT#1,LO%(I):NEXT I
2020 PRINT#1,LU:PRINT#1,T1:PRINT#1,V1:PRINT#1,V2
2030 CLOSE 1:RETURN
2050 REM 6:LOAD
2060 OPEN FOR INPUT AS #1
2070 FOR I=1 TO FO:INPUT#1,LO%(I):NEXT I
2080 INPUT#1,LU,T1,V1,V2
1090 CLOSE #1:RETURN
```

Figura 2.—Subrutinas de "Save" y "Load" para las últimas versiones de BASIC de Microsoft.

Puede haber problemas con las letras minúsculas. Si su ordenador no las tiene o quieren evitar cualquier problema, sencillamente escríbanlo todo en mayúsculas. Si, en cambio, tienen el Commodore 64, se arriesgan a meterse en un problema monstruoso que Commodore arrastra desde hace años: trabajando en modo "sólo mayúsculas" todo va bien, pero pasando a modo "mayúsculas/minúsculas" los códigos de caracteres se cambian en la pantalla (y quedan inalterados en el teclado). Si quieren utilizar el C64 en modo "mayúsculas/minúsculas" deben:

- poner el modo "mayúsculas/minúsculas" con SHIFT/C=
- escribir el programa en minúsculas,
- escribir las REM en minúsculas,
- escribir lo que está entre comillas tal y como aparece en el listado, con las siguientes excepciones:
  - — líneas 1340, 1350: "S" y "N" van en minúscula,
  - — línea 830: "R" va en minúsculas,
  - — líneas 3990-4090: todo en mayúsculas,
  - — líneas 4090, 4230, 4360, 4510: "FD", "FP", "FA", "FO" en minúsculas. Lo mismo en las líneas 1040, 1050, 1060, 1070.

Al introducir el programa tengan cuidado con una serie de cosas:

- — no confundan la letra O (más cuadrada en los listados) con la cifra cero,
- — no omitan los espacios en las cadenas (los que hay entre comillas),
- — no olviden los dos puntos, comas o puntos y comas.

Si quieren, pueden ahorrarse las líneas de comentario (las que empiezan con REM). Cuando hayan acabado, compruébenlo bien todo con la ayuda de otra persona.

¡A jugar!

Cuando tengan el programa metido en la máquina (acuérdense de realizar una copia), pueden dar al RUN y jugar. No les aconsejamos continuar con la lectura del libro antes de haber jugado, pues encontrarían las soluciones de la aventura, estropeándoles la diversión, y tendrían más dificultades para comprender las continuas referencias a la misma contenidas en el programa.

Para los que no sean expertos en aventuras, he aquí las instrucciones fundamentales para un juego de este tipo: Ustedes es-

tán divididos en una doble personalidad, que podríamos llamar el brazo y la mente. Deben darle órdenes a "otro usted mismo" que vive dentro del ordenador y que tiene que afrontar el problema de una nave espacial condenada a un final seguro. Pueden hacerlo con frases del tipo:

MIRA EL COMODIN

(recuerden que deben llamarse siempre de tú) o bien indicando la dirección en la que quieren moverse, por ejemplo:

NORTE

o, más sencillamente:

N

Las posibles direcciones son NORTE, SUR, ESTE, OESTE, ARRIBA y ABAJO; abreviadas quedan, respectivamente, como N, S, E, O, A, B. Pruébenlas en todos los lugares y dibujen un buen plano.
Una acción fundamental es MIRA, que permite acciones tipo

que facilitan información adicional.
Otras acciones esenciales son COGE y DEJA, para manipular los objetos:

Para obtener la lista de los objetos que se poseen, escriban COSA o INVENTARIO.
Si quieren grabar la situación en disco (o casete) para continuar más adelante, escriban:

SAVE

Esta orden también es útil antes de ejecutar una acción peligrosa, para no tener que empezar desde el principio. Si les ocurre algún accidente desagradable, basta escribir después:

LOAD

Para retomar la situación exactamente como estaba en el momento del último SAVE.
Naturalmente, los vocablos citados no agotan todas las posi-

bilidades. Si verdaderamente no encuentran la acción que desean, pueden echar un vistazo a las líneas 3990 a 4090 que contienen las palabras admitidas.
La aventura es muy sencilla de comprender (quitando quizá la acción final, que requiere razonar un poco), pero bastante cuidada y "realista". Esperamos que se diviertan jugando con ella y que la resuelvan sin demasiados problemas. En cualquier caso, no se desesperen: en los próximos capítulos la describimos detalladamente. Claro que estaría mejor que la consiguieran resolver solos...
Este es el programa en versión para el Apple II (para los otros ordenadores, vea el apartado anterior):

```
100 REM ##########################################
110 REM ##  AVENTURA:LA NAVE ESPACIAL CONDENADA ##
120 REM ##########################################
130 REM
140 GOTO 710:REM MAIN
150 REM
160 REM BUSCA P$ EN DICCIONARIO,C=CODIGO (0=AUSENTE)
170 REM
180 I=1:F=FD
190 A=INT((I+F)/2):A$=DZ$(A):IF P$=A$ THEN C=DZ(A):GOTO 240
200 IF P$>A$ THEN I=A+1
210 IF P$<A$ THEN F=A-1
220 IF I<=F THEN 190
230 C=0
240 RETURN
250 REM
260 REM EXTRAE P$ DE IN$[IN],ENCUENTRA CODIGO C,SALTA ART.
270 REM
280 C=0
290 C$=MID$(IN$,IN,1):IF C$=" " OR C$="'" THEN IN=IN+1:
    GOTO 290
300 IF IN>LI THEN P$="":GOTO 360
310 A=IN:REM INICIO
320 C$=MID$(IN$,IN,1)
330 IF C$<>" " AND C$<>"'" AND IN<=LI THEN IN=IN+1:GOTO 320
340 P$=MID$(IN$,A,IN-A):GOSUB 180:REM BUSCA
350 IF C=7 THEN 280:REM ARTICULO
360 RETURN
370 REM
```

```
380 REM BUSCA ACCION A,A=ACCION (0 SI NO ENCONTRADA)
390 REM
400 I=1:F=FA:N=A
410 A=INT((I+A)/2):M=CA(A):IF N=M THEN THEN A=AZ%(A):GOTO 460
420 IF N>M THEN I=A+1
430 IF N<M THEN F=A-1
440 IF I<=F THEN 410
450 A=0
460 RETURN
470 REM
480 REM EJECUTA ACCION A,A=0 SI NO ENCONTRADA
490 REM
500 GOSUB 400:IF A=0 THEN 550
510 IF C2=0 OR A>0 THEN 530:REM SOLO VERBO O NO TEST
520 A=-A:IF OB=0 THEN PRINT "-Aqui no esta ":GOTO 540
530 GOSUB 1710:REM EJECUTA
540 A=1
550 RETURN
560 REM
570 REM ENUMERA OBJETOS EN LUGAR L,CON PREFIJO P$
580 REM
590 FOR I=1 TO FO
600 IF ABS(LO%(I))=L THEN PRINT P$;OB$(I);"."
610 NEXT I:RETURN
620 REM
630 REM OB=INDICE OBJETO C2,0 SI NO PRES. O TRANSP.
640 REM
650 OB=0:FOR I=1 TO FO
660 IF CO%(I)C2 THEN IF ABS (LO%(I))=LU OR LO%(I)=0
    THEN OB=I:I=FO
670 NEXT I:RETURN
680 REM
690 REM PRINCIPAL (PARSER)
700 REM
710 GOSUB 970:REM INICIO
720 GOSUB 1540:REM INTRODUCCION
730 GOSUB 1120:REM <SP>
740 REM
750 REM CICLO DE JUEGO
760 PRINT:PRINT "Estas ";DE$(LU);".":REM DESCRIPCION
```



```
770 L=LU:P$="Veo":GOSUB 590:REM OBJETOS
780 GOSUB 1400
790 PRINT:PRINT
800 IN$="":INPUT "¿Que tengo que hacer?";IN$:
    IF IN$="" THEN 800
810 PRINT:LI=LEN(IN$):IN=1:GOSUB 280:P1$=P$:C1=C
820 IF P1$="" THEN PRINT "¿Y bien?":GOTO 760
830 IF RIGHT$(P1$,1)="R" THEN PRINT
    "Llamame de tu,por favor":GOTO 760
840 IF C1=0 AND P1$<>"" THEN PRINT
    "No conozco el verbo";P1$:GOTO 760
850 GOSUB 280:P2$=P$:C2=C
860 IF C2=0 AND P2$<>"" THEN PRINT
    "No conozco la palabra";P2$:GOTO 760
870 IFC2<>0 THEN GOSUB 650:REM OB=INDICE
880 N1=LU*10000:N2=C1*100
890 A=N1+N2+C2:GOSUB 500:IF A THEN GOTO 760:
    REM VERBO+NOMBRE EN LU
900 IF C2<>0 THEN A=N1+N2+99:GOSUB 500:
    IF A THEN GOTO 760:REM VERBO+X EN LU
910 A=N2+C2:GOSUB 500:IF A THEN GOTO 760:
    REM VERBO+NOMBRE GENERAL
920 IF C2<>0 THEN A=N2+99:GOSUB 500:
    IF A THEN GOTO 760:REM VERBO+X GENERAL
930 PRINT "No entiendo":GOTO 760
940 REM
950 REM INICIO
960 REM
970 DIM DZ$(100),DZ(100):REM DICCIONARIO
980 DIM DE$(30),DI$(30):REM PLANO
990 DIM CA(150),AZ%(150):REM ACCIONES
1000 DIM OB$(50),CO%(50),LO%(50):REM OBJETOS
1010 F$="ASTRO":REM FICHERO SITUACION
1020 REM FD,FP,FA,FO=0
1030 PRINT:PRINT "Un poco de paciencia";
1040 READ A$:IF A$<>"FD" THEN FD=FD+1:DZ$(FD)=A$:
     READ DZ(FD):GOTO 1040
1050 READ A$:IF A$<>"FP" THEN FP=FP+1:DE$(FP)=A$:
     READ DI$(FP):GOTO 1050
1060 READ A$:IF A$<>"FA" THEN FA=FA+1:CA(FA)=VAL(A$):
     READ AZ%(FA):GOTO 1060
```

```
1070 READ A$:IF A$<>"FO" THEN FO=FO+1:OB$(FO)=A$:
     READ CO%(FO),LO%(FO):GOTO 1070
1080 PRINT:PRINT:RETURN
1090 REM
1100 REM <SP> PARA CONTINUAR
1110 REM
1120 PRINT "Pulsa <espacio> para continuar"
1130 GET A$:IF A$<>" " THEN 1130
1140 PRINT:PRINT:RETURN
1150 REM
1160 REM ** FINAL INTERPRETE,PRINCIPIO AVENTURA **
1170 REM
1180 REM MUERTO
1190 REM
1200 GOSUB 1120:REM <SP>
1210 PRINT "*** EL BOLETIN DE LA GALAXIA ***"
1220 PRINT:PRINT "Tragedia cerca de Vega":PRINT
1230 PRINT "La nave espacial Neutronia,en servicio de"
1240 PRINT "pasajeros,con 250 personas a bordo,"
1250 PRINT "ha sido destruida por una violenta"
1260 PRINT "explosion,causada probablemente"
1270 PRINT "por la impericia de su comandante (un"
1280 PRINT "principiante,segun rumores "
1290 PRINT "recojidos por nuestro corresponsal)."
1300 PRINT "El responsable del peor desastre"
1310 PRINT "de la historia galactica vera su memoria"
1320 PRINT "marcada por la tragedia":PRINT
1340 INPUT "¿Quieres seguir jugando?";A$
1350 A$=LEFT$(A$,1):IF A$<>"S" AND A$<>"N" THEN 1330
1360 IF A$="S" THEN RUN
1370 PRINT:PRINT:PRINT "¡¡Adios!!":PRINT:END
1370 REM
1380 REM TIEMPO
1390 REM
1400 T1=T1-1
1410 IF T1<=10 THEN PRINT "Oigo una señal de alarma"
1420 IF T1<=5 THEN PRINT "La temperatura es insoportable"
1430 IF T1<=2 THEN PRINT "La nave es sacudida por vibraciones"
1440 IF T1>0 THEN RETURN
1450 PRINT:PRINT "*** ¡¡Demasiado tarde!! ***":PRINT
1460 PRINT "El reactor esta fuera de control"
1470 PRINT:PRINT "La nave esta desintegrada "
1480 PRINT "en fragmentos minusculos."
1490 PRINT "es increible el silencio"
1500 PRINT "de las explosiones en el vacio":GOTO 1200
1510 REM
1520 REM INTRODUCCION
1530 REM
1540 PRINT:PRINT:PRINT
1550 PRINT "**************************************"
1560 PRINT "* LA NAVE ESPACIAL ESPACIAL CONDENADA *"
1570 PRINT "**************************************":
     PRINT
1580 PRINT "Tumbado en la arena,disfrutas del"
1590 PRINT "dulce calor del sol tropical..."
1600 PRINT:PRINT "Ahora el sol pega mas fuerte,"
1610 PRINT "Estas en pleno desierto y no hay"
1620 PRINT "ni rastro de oasis..."
1630 PRINT:PRINT "Te despiertas sobresaltado en tu"
1640 PRINT "cabina de comandante del Neutronia"
1650 PRINT "Hace calor.Demasiado calor.Tiene"
1660 PRINT "que haber algo que no funciona"
1670 T1=100:LU=6:V1=0:V2=0:PRINT:RETURN
1680 REM
1690 REM EJECUTA ACCION A
1700 ON A GOTO 1780,1830,1920,1970,2000,2060,
     2120,2150,2180,2210
1710 ON A-10 GOTO 2260,2300,2380,2410,2460,2580,
     2640,2700,2740,2800
1730 ON A-20 GOTO 2860,2919,2960,3010,3040,3080,
     3190,3320,3370,3400
1740 ON A-30 GOTO 3440,3520,3570,3640,3800,3860,
     3900,3940
1750 PRINT "ACCION";A:RETURN:REM TEST
1760 REM
1770 REM 1 DIRECCIONES
1780 A=VAL(MID$(DI$(LU),2*C1-1,2))
1790 IF A=0 THEN PRINT "Por alli no puedes ir":RETURN
1800 LU=A:RETURN
1810 REM
```

```
1820 REM 2 COGE
1830 IF LO%(OB)=0 THEN PRINT "Ya hecho":RETURN
1840 IF LO%(OB)<0 THEN PRINT "No es posible":RETURN
1850 IF OB=4 AND LO%(22)=0 THEN PRINT
     "Quitate antes el traje":RETURN
1860 IF O=22 AND LO%(4)=0 THEN PRINT
     "Quitate antes el mono":RETURN
1870 LO%(OB)=0
1880 IF OB=4 OR OB=9 OR OB=11 THEN PRINT
     "Ahora lo tienes puesto":RETURN
1890 PRINT "Hecho":RETURN
1900 RETURN
1910 REM 3 DEJA
1920 IF OB=0 OR LO%(OB)<>0 THEN PRINT
     "No lo tienes":RETURN
1930 IF LU<9 THEN LO%(OB)=LU:PRINT "Hecho":RETURN
1940 LO%(OB)=-99:PRINT "Se ha perdido en el espacio":
     RETURN
1950 REM
1960 REM 4 MIRA
1970 PRINT "No noto nada de particular":RETURN
1980 REM
1990 REM 5 SAVE
2000 D$=CHR$(4):PRINT D$;"OPEN";F$:PRINT D$;"WRITE";F$
2010 FOR I=1 TO FO:PRINT LO%(I):NEXT I
2020 PRINT LU:PRINT T1:PRINT V1:PRINT V2
2030 PRINT D$;"CLOSE":RETURN
2040 REM
2050 REM 6 LOAD
2060 D$=CHR$(4):PRINT D$;"OPEN";F$:PRINT D$;"READ";F$
2070 FOR I=1 TO FO:INPUT LO%(I):NEXT I
2080 INPUT LU,T1,V1,V2
2090 PRINT D$;"CLOSE":RETURN
2100 REM
2110 REM 7 COSA
2120 PRINT "Posees:":L=0:P$="- ":GOTO 590
2130 REM
2140 REM 8
2150 RETURN
2160 REM
2170 REM 9
```



```
2180 RETURN
2190 REM
2200 REM 10
2210 IF LO%(OB)<>0 THEN PRINT "No lo tienes":RETURN
2220 IF NOT(LU>=9 OR (LU=7 AND V2=1)) THEN 1920:REM DEJA
2230 PRINT "¡¡EL AIRE,EL AIRE!! ¡¡¡Aaagh!!!":GOTO 1200
2240 REM
2250 REM 11
2260 PRINT "Es tu mono para actividades"
2270 PRINT "extravehiculares":RETURN
2280 REM
2290 REM 12
2300 PRINT "Esta sin conocimiento y tiene"
2310 PRINT "una abolladura en el casco."
2320 PRINT "probablemente ha sido golpeado"
2330 PRINT "por un pequeño meteorito mientras"
2340 PRINT "reparaba la antena.Por"
2350 PRINT "suerte aun esta vivo":RETURN
2360 REM
2370 REM 13
2380 PRINT "¿No es mejor leerlo?":RETURN
2390 REM
2400 REM 14
2410 PRINT "Es bastante pesado"
2420 PRINT "Probablemente esta"
2430 PRINT "tratado con plomo":RETURN
2440 REM
2450 REM 15
2460 IF LO%(OB)=LU THEN PRINT
     "Cogelo en la mano antes":RETURN
2470 PRINT "-MANUAL DE INSTRUCCIONES DEL-"
2480 PRINT "-    REACTOR POSITRONICO    -"
2490 PRINT "-      MOD.YTREWQ 8421      -":PRINT
2500 PRINT "Para activar el reactor"
2510 PRINT "tirar de la palanca y despues apretar"
2520 PRINT "consecutivamente los pulsadores verde,"
2530 PRINT "amarillo y rojo."
2540 PRINT "Para desctivar el reactor..."
2550 PRINT:PRINT "¡¡Maldicion!! La pagina esta arrancada":
     RETURN
2560 REM
```

```
2570 REM 16
2580 PRINT "-TEMPERARURA REACTOR-":PRINT
2590 PRINT "Marca ";840-T1*4;" grados y esta"
2600 PRINT "subiendo rapidamente.Hay"
2610 PRINT "una marca roja en 800 grados":RETURN
2620 REM
2630 REM 17
2640 PRINT "S.O.S. GALACTICO":PRINT
2650 PRINT "Apretar el pulsador solo"
2660 PRINT "en caso de emergencia"
2670 PRINT "Todo abuso sera castigado":RETURN
2680 REM
2690 REM 18
2700 PRINT "Se enciende brevemente un aviso":PRINT
2710 PRINT "ANTENA EXTERIOR DEFECTUOSA":RETURN
2720 REM
2730 REM 19
2740 PRINT "Esta en la base de una escalerilla"
2750 PRINT "y dice:":PRINT
2760 PRINT "ENTRADA RESERVADA AL"
2770 PRINT "PERSONAL DE A BORDO":RETURN
2780 REM
2790 REM 20
2800 LU=1:IF LO%(20)<>9 THEN RETURN
2810 PRINT "Ojala estuviese aqui el segundo"
2820 PRINT "piloto,es el unico que entiende"
2830 PRINT "los problemas tecnicos":RETURN
2840 REM
2850 REM 21
2860 PRINT "Mejor no despertar a los"
2870 PRINT "pasajeros,podrian"
2880 PRINT "dejarse llevar por el panico":RETURN
2890 REM
2900 REM 22
2910 PRINT "Esta puesto al oeste y dice:":PRINT
2920 PRINT "ATENCION:":PRINT
2930 PRINT "Habitacion despresurizada":RETURN
2940 REM
2950 REM 23
2960 IF LO%(24)<>-99 THEN PRINT "Ya hecho":RETURN
```



```
2970 IF LO%(23)<>0 THEN PRINT
     "Esta cerrado con llave":RETURN
2980 PRINT "Hecho":LO%(24)=LU:LO%(22)=LU:RETURN
2990 REM
3000 REM 24
3010 PRINT "Esta vacio":RETURN
3020 REM
3030 REM 25
3040 PRINT "Dime:aprieta el rojo"
3050 PRINT "o aprieta el verde":RETURN
3060 REM
3070 REM 26
3080 IF V2=1 THEN PRINT "CLICK":RETURN
3090 PRINT "La pared al este se cierra"
3100 PRINT "La pared al oeste se abre"
3110 PRINT "hacia el espacio exterior"
3120 PRINT "El aire sale silbando"
3130 IF LO%(4)<>0 OR LO%(11)<>0 THEN PRINT:
     PRINT "¡¡Aaagh!!":GOTO 1200
3140 FOR I=1 TO FO
3150 IF LO%(I)=LU THEN PRINT
     OB$(I);"Se pierde en el espacio":LO%(I)=-99
3160 NEXT I:V2=1:RETURN
3170 REM
3180 REM 27
3190 IF V2=0 THEN PRINT "CLICK":RETURN
3200 PRINT "La pared al oeste se cierra"
3210 PRINT "La pared al este se abre"
3220 PRINT "hacia el pasillo"
3230 PRINT "El aire vuelve a entrar silbando":V2=0
3240 IF LO%(20)<>0 OR LO%(23)<>-99 THEN RETURN
3250 PRINT:PRINT "El segundo piloto revive,se da"
3260 PRINT "cuenta enseguida de la situacion y dice:"
3270 PRINT:PRINT "Pronto,¡para el reactor!"
3280 PRINT "He aqui la llave de mi..."
3290 PRINT "Despues pierde de nuevo el sentido":
     LO%(23)=LU:RETURN
3300 REM
3310 REM 28
3320 PRINT "¡La emergencia acecha!"
3330 PRINT "Lleva siempre contigo"
```

```
3340 PRINT "El manual del reactor":RETURN
3350 REM
3360 REM 29
3370 PRINT "El tecnico es el segundo piloto":RETURN
3380 REM
3390 REM 30
3400 PRINT "Dime:aprieta el rojo,aprieta"
3410 PRINT "el verde o aprieta el amarillo":RETURN
4320 REM
3430 REM 31
3440 PRINT "CLICK":IF V1<>0 THEN 3830
3450 PRINT "Una tuberia pierde ligeramente"
3460 PRINT "por un empalme (probablemente por"
3470 PRINT "exceso de presion).Las gotas caen sobre el"
3480 PRINT "cuadro de control,cerca de ti"
3490 V1=1:RETURN
3500 REM
3510 REM 32
3520 PRINT "CLICK":IF V1<>1 THEN 3830
3530 V1=2:IF LO%(22)<>0 THEN PRINT
     "No te encuentras muy bien"
3540 RETURN
3550 REM
3560 REM 33
3570 PRINT "CLICK":IF V1<>2 THEN 3830
3580 V1=3:IFLO%(22)=0 THEN RETURN
3590 PRINT "Temo que hayas absorbido demasiadas"
3600 PRINT "radiaciones.Ahora estas realmente"
3610 PRINT "mal.Pierdes el conocimiento...":GOTO 1200
3620 REM
3630 REM 34
3640 PRINT "CLANK":IF V1<>3 THEN 3830
3650 GOSUB 1120:REM <SP>
3660 PRINT "*** EL BOLETIN DE LA GALAXIA ***"
3670 PRINT:PRINT "COMANDANTE SALVA NAVE ESPACIAL"
3680 PRINT "La nave espacial Neutronia,en servicio"
3690 PRINT "de pasajeros,con 250 personas a bordo,"
3700 PRINT "ha sido salvada de la segura destruccion"
3710 PRINT "gracias al valor y a la sangre fria"
3720 PRINT "del comandante,que ha conseguido"
3730 PRINT "desactivar el reactor enloquecido."
```



```
3740 PRINT "Su nombre sera recordado"
3750 PRINT "siempre entre los heroes de nuestra"
3760 PRINT "flota galactica":PRINT
3770 PRINT:PRINT "¡¡FELICIDADES!!":PRINT:END
3780 REM
3790 REM 35
3800 PRINT "CLUNK":GOTO 3830
3810 REM
3820 REM COMUNE
3830 V1=0:T1=INT(T1/2):RETURN
3840 REM
3850 REM 36
3860 PRINT "Parece deteriorada"
3870 PRINT "Por un gran meteorito":RETURN
3880 REM
3890 REM 37
3900 IF V2=0 THEN LU=4:RETURN
3910 GOTO 1780:REM DIRECCION
3920 REM
3830 REM 38
3940 IF V2=1 THEN LU=11:RETURN
3950 GOTO 1780:REM DIRECCION
3960 REM
3970 REM DICCIONARIO
3980 REM
3990 DATA "A",5,"ABAJO",6,"ABRE",22,
     "AMARILLO",63,"ANTENA",69,
4000 DATA "APRIETA",26,"ARMARIO",67,"ARREGLA",
     27,"ARRIBA",5,
4010 DATA "B",6,"BAJA",6,"CAMA",68,"CARTEL",60,
     "CASCO",50,"COGE",8,
4020 DATA "COSA",13,"DEJA",9,"E",3,"EL",7,
4030 DATA "EMPUJA",24,"ESTE",3,"ETIQUETA",70,
     "INDICADOR",66,"INVENTARIO",13,
4040 DATA "LA",7,"LEE",25,"LLAVE",54,"LO",
     7,"LOAD",12,
4050 DATA "MANUAL",55,"MIRA",10,"MONO",51,"N",1,
     "NORTE",1,"O",4,
4060 DATA "OESTE",4,"PALANCA",65,"PONTE",20,
     "PULSADOR",61,"QUITA",21,
```

```
4070 DATA "REPARA",27,"ROJO",64,"S",2,"SAVE",11,
     "SEGUNDO",53,
4080 DATA "SUBE",5,"SUR",2,"TECLA",61,"TIRA",23,
     "TRAJE",52,
4090 DATA "VERDE",62,"VISTA",20,"W",4,"FD"
4100 REM
4110 REM PLANO
4120 REM
4130 DATA "En la cabina de mando","000000000002"
4140 DATA "En un extremo del pasillo",
     "000300050100"
4150 DATA "En el pasillo","020400060000"
4160 DATA "En un extremo del pasillo",
     "030000070008"
4170 DATA "En la cabina del segundo piloto",
     "000002000000"
4180 DATA "En tu cabina","000003000000"
4190 DATA "En el compartimento estanco",
     "000000000000"
4200 DATA "En la sala del reactor","000000000400"
4210 DATA "En el exterior,a proa de la nave",
     "001000000000"
4220 DATA "En el exterior de la nave",091100000000"
4230 DATA "En el exterior,a popa de la nave",
     "100007000000","FP"
4240 REM
4250 REM ACCIONES
4260 REM
4270 DATA 100,1,200,1,300,1,400,1,500,1,600,1,899,-2
4280 DATA 950,-10,951,-10,999,3,1051,-11,
     1052,-14,1053,-12
4290 DATA 1055,-13,1060,-13,1099,-4,1100,
     5,1200,6,1300,7
4300 DATA 2050,-2,2051,-2,2052,-2,2150,-10,2151,-10,
     2152,-3,2255,-15
4310 DATA 2555,-15,2769,-29,6550,-10,6551,-10,6552,-3
4320 DATA 11066,16,12570,17,12661,18,20300,21,20500,
     20,22560,19
4330 DATA 30300,21,40300,21,42560,22,52267,23,62267,24
4340 DATA 70300,37,70400,38
```



```
4350 DATA 72661,25,72662,27,72664,26,81066,16,82365,
     35,82465,34
4360 DATA 82560,28,82661,30,82662,33,82663,32,82664,
     31,91069,36,"FA"
4370 REM
4380 REM OBJETOS
4390 REM
4400 DATA "Un indicador",66,-1,"Un pulsador",61,-1
4410 DATA "Una etiqueta",70,-1,"Un mono",51,1
4420 DATA "Un cartel blanco",60,-2,
     "Un cartel amarillo",60,-4
4430 DATA "Una cama",68,-5,"Un armario",67,-5
4440 DATA "Una cama",68,-6,"Un armario",67,-6
4450 DATA "Un casco",50,6,"Un pulsador rojo",61,-7
4460 DATA "Un pulsador verde",61,-7,
     "Un indicador",66,-8
4470 DATA "Una palanca",65,-8,"Un pulsador rojo",61,-8
4480 DATA "Un pulsador verde",61,-8,
     "Un cartel rojo",60,-8
4490 DATA "Un pulsador amarillo",61,-8,
     "El segundo piloto",53,9
4500 DATA "Una antena parabolica",69,-9,
     "Un traje",52,-99
4510 DATA "Una llave",54,-99,"Un manual",55,-99,"fo"
```

CAPITULO III

LA PUESTA EN ESCENA



tención: este capítulo contiene la solución de la aventura. Si lo leen antes de haber jugado se perderán toda la diversión.

La idea

Realizar una aventura requiere fantasía y técnica; igual que para escribir una novela, pintar un cuadro o cualquier otra forma de expresión, ambos componentes son fundamentales para realizar un buen trabajo.

Está claro que nosotros sólo podemos proporcionarles los instrumentos técnicos, pero no la fantasía o la creatividad, aunque podemos serles útiles en la organización de las ideas y en la fase inicial de creación del juego, enseñándoles cómo se gestó y nació "La nave espacial condenada".

Lo primero que se necesita es una idea, una ambientación: ¿dónde transcurre la aventura? ¿Cuál es el argumento de la historia?

Era un sábado por la mañana de un crudo invierno, con 85 centímetros de nieve rodeando la casa...; pero no divaguemos, esta es otra aventura. Era sábado por la mañana, decíamos, y estábamos discutiendo sobre cuál podría ser el tema de una pequeña aventura para ponerlo de ejemplo en este libro. La primera idea fue la búsqueda de un tesoro en una isla del Caribe, en las antiguas rutas de los piratas. Escribimos unas cuantas notas; las ideas no nos faltaban, incluso algunas eran tan buenas que decidíamos guar-

darlas para otra futura aventura (esto sucede a veces). Pero luego pensamos que la historia de la isla no nos convencía; entonces ¿qué ambiente utilizar?

No queríamos una ambientación "gótica" (tipo "Aventura en el castillo") porque no se prestaba demasiado para aventuras de pequeño tamaño, donde el entorno no se puede desarrollar más allá de un determinado punto. También era necesario motivar al aventurero: no hay nada peor que una aventura donde el jugador se siente "fuera" del juego en lugar de "vivirlo" realmente en primera persona. ¿Un reactor nuclear enloquecido? Demasiado simple. ¿Un ordenador malicioso a bordo de una nave espacial (como el HAL 9000 en la célebre "2001, Odisea en el espacio")? Más interesante, pero difícil de desarrollar en poco espacio. Quitemos el ordenador y pongamos el reactor enloquecido a bordo de la nave espacial. Casi hemos llegado. Añadamos un tiempo límite y 250 pasajeros, ignorantes de lo que ocurre, que hay que salvar (además de nosotros mismos). Sí, así está bien.

El guión

Ahora que ya tenemos la idea de partida, intentemos desarrollar un guión coherente. El protagonista es el comandante de una nave espacial que tendrá que conseguir detener a tiempo el reactor. Naturalmente, tendrá que hacerlo todo él solo: no puede descargar en nadie ninguna parte de su propia responsabilidad pues, si no, el juego pierde interés y se convierte en un "pasarse la pelota" de unos a otros. Entonces, el comandante está solo mientras los pasajeros están, pongámoslo así, durmiendo tranquilamente.

Si bien puede ser cierto que las modernas naves espaciales requieran poco personal, es poco plausible que sea suficiente con una sola persona. Si en el juego se comprueba que el comandante no es capaz de parar el reactor, es necesario que haya otra persona capaz de hacerlo. Por otra parte, esa otra persona no debe de estar muy a mano, porque si no el problema se resolvería fácilmente. Introducimos entonces la figura de un segundo piloto, con funciones técnicas, pero haciendo que no esté fácilmente disponible. En principio no se sabe qué pasa con él: podría haberse suicidado (típico síntoma de depresión espacial), haber sido víctima de accidente (quizá a causa del reactor) o, simplemente, andar por ahí despistado.

En toda aventura espacial que se precie no puede faltar una salida al espacio exterior. Por lo tanto deberemos de admitirla de alguna manera. He aquí cómo: el segundo piloto se encuentra en el exterior de la nave espacial y sólo rescatándolo se podrá obtener una información esencial. En ese momento aún puede estar vivo: así su rescate formará parte de la misión. Una vez rescatado, recobra el conocimiento (pero sólo por un momento) y nos entrega algo fundamental. ¿Qué? Veamos... El libro de instrucciones que sirve para desactivar el reactor. No, alarguemos algo más el asunto: la llave de su armario, que contiene el manual de instrucciones y un traje antirradiaciones necesarios para trabajar en el reactor. Además, para dificultar la solución, tampoco estará muy claro qué es lo que nos da.

A propósito, ¿por qué el segundo piloto estaba en el exterior? Utilizaremos una idea de "2001": estaba arreglando la antena de comunicaciones (la nave espacial también estará, por tanto, aislada del resto del universo).

Con esto el guión está completo en sus líneas generales: es necesario equiparse para salir al vacío (mono y casco), salir al exterior, rescatar al segundo piloto, conseguir la llave, abrir su armario, coger el manual y el traje, leer el manual y desactivar el reactor.

Parece factible. Los detalles los añadiremos poco a poco.

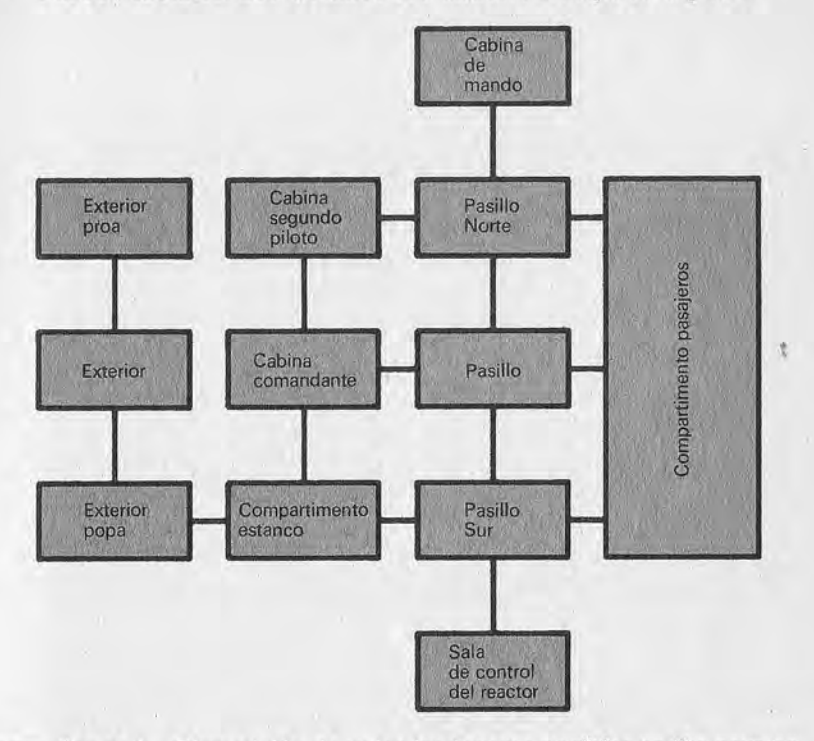


Figura 1.—Disposición de todos los lugares posibles en "La nave espacial condenada".

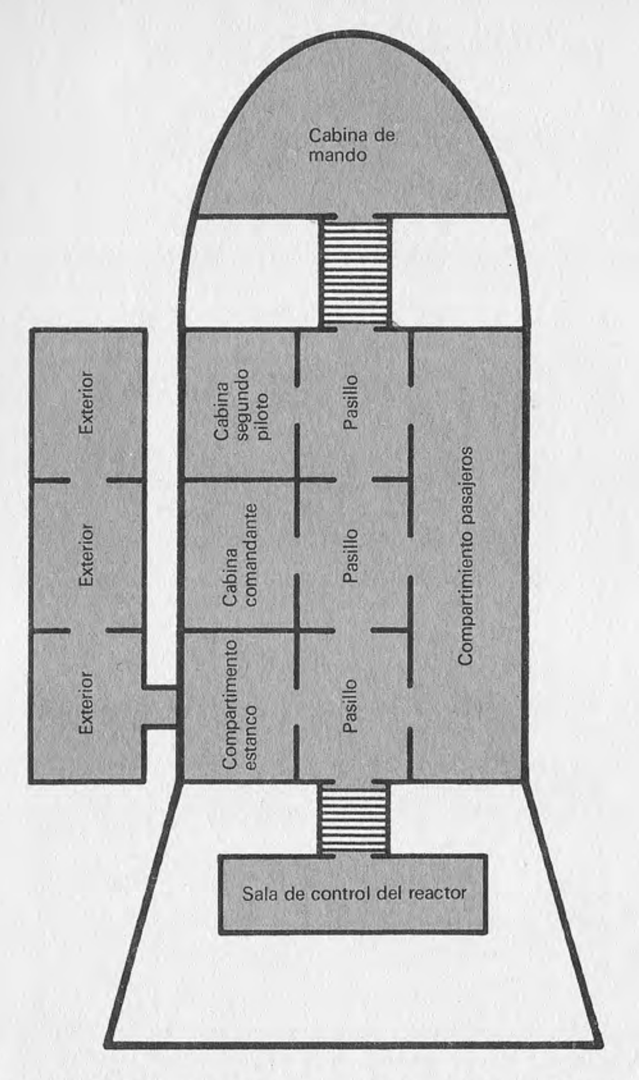


Figura 2.—Plano de la nave espacial de pasajeros "Neutronia".

El plano de la nave

Nos falta establecer la disposición de las diversas estancias que configuran la nave espacial. Dibujemos un plano, que nos sirva de guía, con el mismo sistema ya usado en otras ocasiones: cua-

drados adosados con líneas que indican los caminos y direcciones posibles. Adoptando un criterio convencional el Norte está arriba, el Sur abajo, el Este a la derecha y el Oeste a la izquierda (como en los mapas geográficos). Para indicar un movimiento hacia arriba, hacemos partir la línea desde el ángulo superior derecho y si es hacia abajo, desde el ángulo inferior izquierdo.

La figura 1 muestra la disposición de los locales, dibujada con este sistema. Nótese el compartimento estanco, necesario para salir al exterior sin que se escape el aire de la nave espacial.

Parece que está bien proyectado, podemos pasarlo a limpio. La figura 2 muestra la versión definitiva del mismo plano. Las proporciones reales no se han respetado, y hemos omitido algunos locales de menor interés, pero las severas Normas De Seguridad Galáctica del 2312 nos prohíben proporcionar más información, aunque se trate de una nave espacial civil.

Aparte de los detalles, el diseño del proyecto de la aventura ha terminado: no queda más que ponerse manos a la obra. Como información adicional les diremos que, aparte del tiempo para pasarlo a limpio, hemos empleado poco más de un par de horas.

CAPITULO IV

CÓMO NO SE ESCRIBE UNA AVENTURA

De la idea al programa



Ahora que disponemos ya de un guión aceptado, podemos empezar a escribir el programa. ¿Cómo se realiza un programa de este tipo? Es el mismo problema con que nos encontramos en 1981 cuando, después de haber jugado a "Apple Adventure", decidimos escribir una aventura por nuestra cuenta. En aquel tiempo estábamos totalmente desinformados y no leíamos siquiera las principales revistas del sector, especialmente las americanas.

Una pena, porque habríamos podido, por lo menos, conocer lo que otros habían hecho o estaban haciendo. Esto se explica porque en aquellos tiempos éramos simplemente aficionados en este tema y esas publicaciones no circulaban ni siquiera en los ambientes supuestamente más cualificados (por ejemplo, la Universidad), donde todavía estaba vigente el culto al ordenador-dinosaurio (a base de tarjetas perforadas y con largas colas para poder usarlo).

No tuvimos más remedio que ir aprendiendo poco a poco y arreglárnoslas por nosotros mismos. Por suerte teníamos bastante práctica en programación, de forma que al final acabamos sabiendo un poco. Después de un año de trabajo en los momentos libres, logramos escribir un programa al que incluso hoy, como profesionales, todavía podemos mirar sin avergonzarnos. Dadas las circunstancias, es una gran satisfacción. "La nave espacial condenada" deriva directamente de "Aventura en el castillo", que es el programa que acabamos de mencionar, con muchas ampliaciones y mejoras derivadas de un estudio del que hablaremos posteriormente.

La mejora fundamental, por lo menos desde nuestro punto de vista, consiste en el lenguaje utilizado: el programa "Aventura en el castillo" (en su primera versión) estaba escrito mitad en BASIC mitad en Ensamblador del 6502 (el lenguaje máquina del Apple), con todos los trucos posibles para ahorrar memoria y aumentar la velocidad de ejecución. Esto significa que estaba ligado indisolublemente al modelo específico de ordenador en el que había sido proyectado. El programa de "La nave espacial condenada" está escrito todo en BASIC, y en un BASIC "limpio". Por lo tanto es fácilmente transportable, es decir, puede funcionar con mínimas modificaciones en ordenadores tan distintos entre sí como un IBM PC y un Commodre 64. En cambio, no se presta demasiado a la realización de aventuras de grandes dimensiones: no se puede tener todo. Las simplificaciones que hemos introducido (y que disminuyen un poco la "inteligencia" del programa) han sido hechas en gran parte para concentrar las explicaciones en los aspectos esenciales y no distraer con las florituras del programa.

Planteamientos equivocados

Decíamos que aún estábamos satisfechos con el planteamiento de nuestro primer programa de aventuras. Lo estamos porque, mirando a nuestro alrededor, vemos que la gran mayoría de las aventuras existentes están construidas con técnicas mucho más primitivas y laboriosas. También en algunos libros que enseñan "cómo escribir una aventura", hay planteamientos que obligan al programador a un trabajo excesivamente largo, gran parte del cual podría haber sido tranquilamente reducido por un programa mejor pensado y más flexible.

Esta es la esencia misma de la programación de los ordenadores: escribir un programa que resuelva no sólo un problema, sino un conjunto de problemas parecidos modelables según se varíen los datos introducidos. No tendría sentido escribir, por ejemplo, un programa para sumar 1 + 1, otro para sumar 1 + 2, otro distinto para sumar 2 + 2, y así sucesivamente: un sólo programa puede resolver todos estos problemas. Parece obvio en este caso, pero al aumentar la complejidad y variabilidad del problema, el asunto es cada vez menos fácil y requiere un estudio más detallado. La tentación de escribir un programa que resuelva sólo el problema actual es cada vez mayor. Por ejemplo, hemos visto un juego de aventuras escrito más o menos de este modo:

```
500 REM HABITACION BLANCA
510 PRINT "ESTAS EN LA HABITACION BLANCA.HAY UN ARMARIO"
520 INPUT A$
```



```
530 IF A$="N" OR A$="NORTE" THEN GOTO 610:REM ROJA
540 IF A$="E" OR A$="ESTE" THEN GOTO 710:REM VERDE
550 PRINT "NO ENTIENDO":GOTO 510
...
600 REM HABITACION ROJA
610 PRINT "ESTAS EN LA HABITACION ROJA.HAY UN ARMARIO"
620 INPUT A$
630 IF A$="S" OR A$="SUR" THEN 510:REM BLANCA
640 IF A$="ABRE EL ARMARIO" THEN PRINT "ESTA VACIO":GOTO 610
650 PRINT "NO LO ENTIENDO":GOTO 610
...
```

¿Han entendido cómo funciona? Para cada lugar existe el correspondiente grupo de instrucciones dentro de las cuales el programa continúa girando hasta que el aventurero no decide cambiar de lugar. Por ejemplo, desde las líneas 500 a la 550 se hace referencia a un lugar llamado "Habitación blanca". La línea 510 describe el lugar y la 520 espera las órdenes del jugador (típicamente figurará una subrutina en lugar de una instrucción INPUT, pero el concepto es el mismo). Si el jugador ha escrito "N" o "NORTE", ambas palabras son válidas para indicar un desplazamiento hacia el norte, el programa salta al grupo de instrucciones referentes al lugar que se encuentra al norte de la habiación blanca, en el ejemplo, una hipotética habitación roja (línea 610). Si ha escrito "E" o "ESTE", salta a las instrucciones referentes a la "Habitación verde", y así sucesivamente. Si la orden no está entre las previstas en aquella habitación, la línea 550 vuelve al principio del bucle (510).

Cada habitación (lugar) tiene su propio ciclo de instrucciones, dentro del cual se admiten sólo determinadas palabras o frases. Por ejemplo, en el ciclo de la habitación roja (líneas 610-650) se admiten "N" (o "NORTE"), que vuelve al ciclo de la habitación blanca y, "ABRE EL ARMARIO", que imprime "ESTA VACIO" y vuelve al principio del ciclo. La misma explicación vale para todos los lugares en los que el protagonista se puede mover. Es una técnica simple y comprensible.

Un planteamiento de este tipo hay que evitarlo. ¿Por qué? Vamos a ver una lista de buenos motivos para hacerlo:

- el control de las direcciones posibles (IF A$ = "N" or A$ = "NORTE"...) se repite en cada bucle de lugar;
- las frases válidas se deben escribir exactamente como estén previstas en el programa. Si la frase "ABRE EL ARMARIO" se escribe con un espacio al principio o al final o entre "ABRE" y "ARMARIO", se obtiene un "NO ENTIENDO". Lo mismo sucede si el artículo no está bien colocado;

- hay notables problemas con las frases válidas para todos los lugares, por ejemplo, las acciones sobre un objeto transportable;
- si deseamos escribir otra aventura se deberá volver a escribir el programa desde el principio hasta el final.

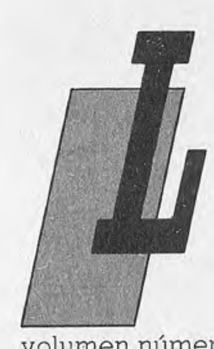
Podríamos continuar, pero nos parece suficiente. En esencia, es necesario un gran trabajo para obtener un programa poco satisfactorio y que ocupa inútilmente mucha memoria. Peor aún, el juego se presenta con un comportamiento decididamente "estúpido", en el sentido de que acepta exclusivamente las frases previstas en el lugar en el que se encuentra el aventurero (que corresponde a un "lugar" físico del programa, entendido como un grupo de instrucciones).

Lo que les hemos presentado es un ejemplo límite (desgraciadamente, real) pero gran parte de las aventuras actuales presentan problemas debidos a un erróneo planteamiento del programa, que se traduce posteriormente en una pérdida de realismo y, por lo tanto, de interés del jugador. Comentario típico: —Pero, esta máquina estúpida ¿no entiende nada?— Ya la ilusión de vivir una aventura fantástica se ha perdido, junto con los esfuerzos del autor.

Está claro que la calidad de una aventura depende de quién la haya escrito, pero también está claro que la fantasía no es suficiente: sin las técnicas apropiadas es difícil realizar un juego que sea capaz de "capturar" a los jugadores proporcionando un nivel aceptable de simulación. En los próximos capítulos demostraremos cómo es posible obtener esto con una estructura compacta, fácil de usar y reutilizable para otras aventuras.

CAPITULO V

POSICIÓN Y MOVIMIENTOS



La mejor solución no es un programa que admite las distintas instrucciones según el lugar, como hemos mostrado (poniéndola como ejemplo negativo) en el capítulo anterior, sino la consistente en un programa que ejecute siempre el mismo ciclo fundamental, cambiando sólo el valor de algunas variables y, consecuentemente, los mensajes impresos en la pantalla. Un ejemplo de este planteamiento ha sido presentado por el volumen número 9 ("Programando como es debido") de esta misma colección.

Una subrutina universal

Para quienes no lo hubiesen leído y también, ¿por qué no?, para los demás, proponemos un ejemplo parecido: un mini-programa o, mejor dicho, una subrutina (es decir, una parte de programa) que imprime la descripción de la habitación en la que se encuentra el aventurero y que le permite moverse con el sistema habitual, o sea, escribiendo la dirección en la que quiere moverse. He aquí la subrutina:

```
100 REM INICIA VARIABLES
...
200 REM CICLO
210 PRINT DE$(LU):REM DESCRIPCION
220 INPUT A$:D=0
```

```
230 IF A$="N" OR A$="NORTE" THEN D=1
240 IF A$="S" OR A$="SUR" THEN D=2
250 IF A$="E" OR A$="ESTE" THEN D=3
260 IF A$="O" OR A$="OESTE" THEN D=4
270 IF A$="A" OR A$="ARRIBA" THEN D=5
280 IF A$="B" OR A$="ABAJO" THEN D=6
290 IF D=0 THEN PRINT "NO ENCONTRADO":GOTO 210
300 NL=PL(LU,D):REM NUEVO LUGAR
310 IF NL=0 THEN PRINT "NO PUEDES":GOTO 210
320 LU=NL:GOTO 210:REM MUEVE
```

Esto es todo. Si al principio (entre las líneas 100 y 200) introducimos los datos apropiados en los elementos de las matrices DE$ y PL, esta subrutina, de 13 líneas, hará todo el trabajo necesario para controlar los movimientos del aventurero sobre un terreno de cualquier complejidad o dimensión. ¿No se lo creen? Pasemos enseguida a una demostración práctica.

Supongamos que nuestro mapa esté compuesto por tres lugares, llamados "Habitación roja", "Habitación blanca" y "Habitación verde", dispuestos como en la figura 1. La figura 2 muestra cómo debe de ser el contenido de las dos matrices DE$ y PL. La primera, de tipo alfanumérico (cadena), contiene sencillamente las descripciones de los lugares en orden numérico. Por lo tanto DE$(1) es la descripción del lugar 1, DE$(2) es la descripción del

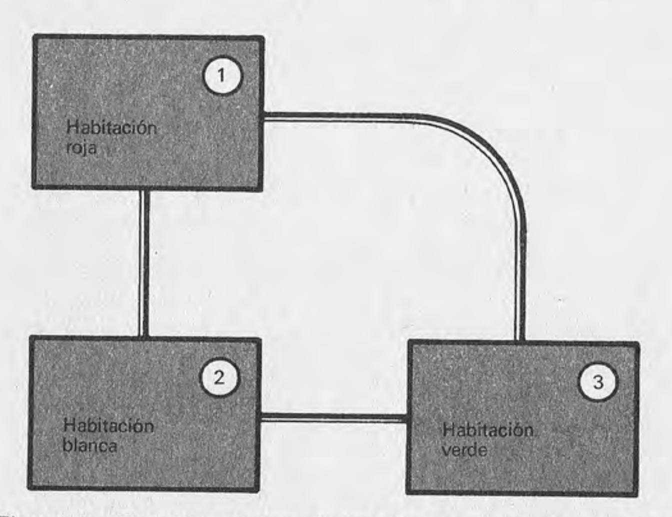
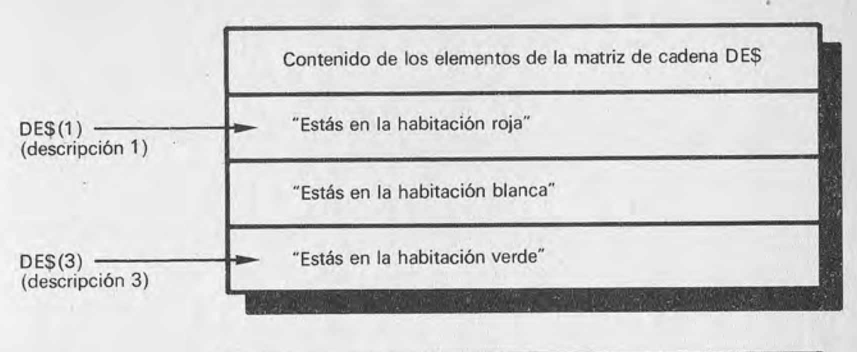
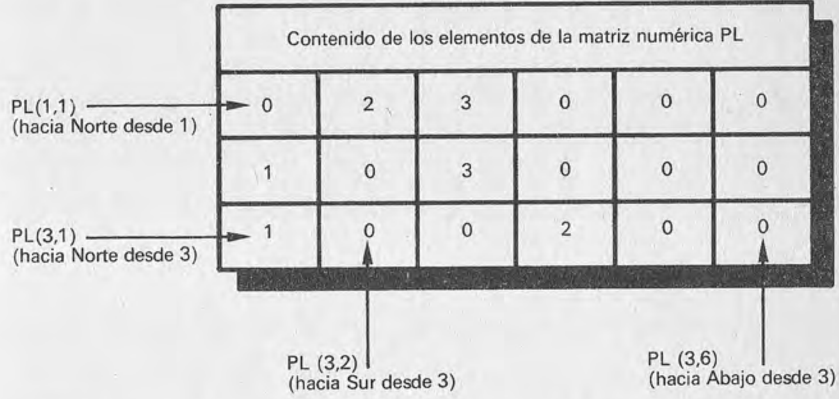


Figura 1.—Ejemplo mínimo de un plano para diseñar aventuras.





0	2	3	0	0	0
1	0	3	0	0	0
1	0	0	2	0	0

Figura 2.—Matrices esenciales del programa y su contenido.

lugar 2 y DE$(3) la descripción del lugar 3. Como habrán deducido, los lugares van numerados a partir de 1.

Volvamos por un momento al programa y precisamente a la línea 210. Esta imprime el contenido de DE$(LU), es decir la descripción del lugar número LU. LU contiene, por lo tanto, el número del lugar actual (o sea, aquel en el que se encuentra el aventurero). Supongamos que hemos puesto LU = 1 al principio del programa: esto significa que el personaje se encuentra en el lugar 1; así la línea 210 imprime DE$(1), es decir, "Estás en la habitación roja" que, efectivamente, es la descripción del lugar 1. Llamamos DE a la matriz por "DEscripción"; siempre es mejor utilizar para las variables nombres que ayuden a recordar su significado.

Numeramos los lugares:

- 1 Habitación roja
- 2 Habitación blanca
- 3 Habitación verde

Llegados aquí, habrán comprendido que para desplazar al aventurero, o sea, para moverse de lugar es suficiente con cambiar el valor de LU. Si LU vale 2, la línea 210 imprime "Estás en la habitación blanca" y si vale 3 imprime "Estás en la habitación verde". El jugador tiene la sensación de que cambia de lugar, pero para el programa cambia simplemente el valor de una variable, precisamente LU. Las instrucciones ejecutadas permanecen invariables.

De acuerdo, pero ¿cómo hacemos para cambiar LU según la palabra escrita por el jugador, de forma que los desplazamientos sean coherentes con nuestro plano? Es muy sencillo: a cada lugar se le han asignado seis números, correspondientes a las seis direcciones, en el orden establecido por nosotros; establezcamos, por ejemplo, el orden Norte, Sur, Este, Oeste, Arriba, Abajo. Estos números están contenidos en una línea de la matriz, o array, de dos dimensiones, PL (PL está por "plano").

Refiriéndonos siempre a la figura 2, los seis números contenidos en PL(1,1), PL(1,2) PL(1,3), PL(1,4), PL(1,5), PL(1,6) son los números de los lugares a los que se llega desplazándose desde el lugar 1 en la dirección indicada.

Númeremos las direcciones:

- Norte = 1
- Sur = 2
- Este = 3
- Oeste = 4
- Arriba = 5
- Abajo = 6

Si desde el lugar 1 el jugador decide moverse en la dirección 3 (hacia el este), el elemento PL (lugar, dirección), o sea PL(1,3) contiene el número del nuevo lugar, en este caso 3 (habitación verde). Si ponemos LU igual a este número, habremos desplazado el personaje al nuevo lugar.

El desplazamiento, por supuesto, no siempre es posible. Para indicar la imposibilidad del movimiento en una cierta dirección desde un determinado lugar, ponemos un cero en el elemento correspondiente. Por ejemplo, desde el lugar 1 no se puede ir hacia el oeste, por eso PL(1,4) contiene un cero.

Resumiendo: cada línea de la matriz PL corresponde a un lugar, y cada columna a una dirección. El cruce entre línea y columna indica si es posible el desplazamiento desde aquel lugar en aquella dirección y, en caso afirmativo, el número del lugar al que se llega. Si LU es el lugar actual y D el número correspondiente a la dirección elegida, PL(LU,D) es el nuevo lugar si el movimiento es posible, o cero si no lo es.

Volvamos a examinar el programa. La línea 220 espera la orden del jugador y pone D a cero (ninguna dirección introducida). Las líneas desde la 230 a la 280 sirven para poner en D el código (número) de la dirección escrita, sin ninguna pretensión (por ahora) de emplar un método más eficiente. Por ejemplo, si el jugador ha escrito "ESTE" la línea 250 ejecuta un D = 3 (código de "Este"). Si la palabra introducida no corresponde a ninguna de las direcciones posibles, D permanece a cero.

La línea 290 comprueba si ha sido introducida una dirección válida. En caso contrario (D vale cero), imprime "NO ENTIENDO" y nos envía al comienzo del bucle.

La línea 300 consulta el plano contenido en la matriz PL para conocer el lugar adyacente al actual (LU) en la dirección indicada (D), y pone en NL el número del nuevo lugar. No lo pasa inmediatamente a LU porque podría ser cero (desplazamiento imposible) y, en este caso, se perdería el valor de LU.

Si el desplazamiento no es válido, la línea 310 imprime "NO PUEDES" y vuelve al comienzo del ciclo (LU no ha cambiado); si lo es, la línea 320 cambia el valor de LU, desplazando así al personaje al nuevo lugar.

Programa y datos

El concepto fundamental que está en la base de lo visto es éste: cambiamos los datos, pero no cambiamos el programa. Efectivamente, si se quiere representar un plano completamente diferente, basta cambiar el contenido de la matriz de descripciones DE$ y de la matriz de conexiones PL, pero el programa permanece igual: el trabajo se ha hecho una sola vez. Como ventaja secundaria, pero no por ello menos importante, el programa ocupa mucho menos espacio.

Un programa de este tipo se llama "orientado por tablas", ya que su comportamiento depende exclusivamente de los datos contenidos en una tabla apropiada, en este caso las matrices PL y DE$. También se llama "máquina de estados variables", en cuanto que la diferencia entre una situación y otra (entre un lugar y el otro) depende exclusivamente del estado (valor) de una o más variables; en nuestro caso solamente de LU (los puristas nos perdonarán esta descripción no demasiado formal). El programa ejecuta siempre en el mismo bucle, dondequiera que se halle el aventurero.

Nos queda el problema de cómo introducir los valores en las tablas, o sea, en las matrices DE$ y PL.

Se podría hacer de esta manera:

```
110 DE$(1)="ESTAS EN LA HABITACION ROJA"
120 DE$(2)="ESTAS EN LA HABITACION BLANCA"
130 DE$(3)="ESTAS EN LA HABITACION VERDE"
140 PL(1,1)=0
150 PL(1,2)=2
160 PL(1,3)=3
...
```

En seguida se ve que ésta no es la solución más práctica: se desperdicia mucha memoria para ejecutar instrucciones repetitivas (piensen en un plano con un centenar de habitaciones). Hay una técnica decididamente mejor, basada en una comodísima instrucción del BASIC que, por otra parte, falta en algunos lenguajes "más evolucionados": la instrucción DATA, que permite introducir en el programa una lista de datos numéricos o de cadenas. Por lo tanto, podemos escribir nuestras dos tablas de esta manera:

```
300 DATA "Estas en la habitacion roja"
310 DATA 0,2,3,,0,0,0
320 DATA "Estas en la habitacion blanca"
330 DATA 1,0,3,0,0,0
340 DATA "Estas en la habitacion verde"
350 DATA 1,0,0,2,0,0
```

A la descripción de cada lugar, le siguen inmediatamente los pasos posibles en las seis direcciones. Para leer estos datos utilizamos la instrucción READ. Equivale a la INPUT, pero lee los valores de la lista de DATA en lugar de por el teclado (a partir del punto al que había llegado la READ anterior).

He aquí a lo que se reduce la inicialización (preparación inicial) de las variables para el programa que hemos mostrado anteriormente:

```
110 NL=3:REM NUMERO LUGARES
120 DIM DE$(NL),PL(NL,6)
130 FOR LU=1 TO NL
140 READ DE$(LU)
150 FOR D=1 TO 6:READ PL(LU,D):NEXT D
160 NEXT LU
170 LU=1
```

Nótese que la línea 110 especifica el número de lugares, que es un valor fijo, o sea una constante. Es una buena regla indicar

las constantes una sola vez al principio del programa, en vez de esparcirlas por todos sitios. Así, las eventuales (en realidad seguras) modificaciones son mucho más simples y es menos fácil cometer errores. Sería aún más inteligente hacer leer el número de lugares de los DATA con un READ NL: el programa permanecería inalterado también al variar el número de lugares.

La línea 120 establece las dimensiones de nuestras tablas, o sea, las dos matrices DE$ y PL. En muchos BASIC es tono es necesario si las dimensiones no superan el 10, como en nuestro caso, pero si se indican tampoco importa. El bucle abierto en la línea 130 y cerrado en la 160 se repite para cada lugar; tres veces, por lo tanto en el ejemplo. La línea 140 lee la descripción del lugar en el elemento correspondiente de la matriz de cadena DE$, mientras el bucle en la 150 lee el plano de las direcciones en la matriz bidimensional PL. Finalmente, la línea 160 establece que el personaje inicia el juego en el lugar 1.

CAPITULO VI

EL ORDENADOR ENTIENDE CASTELLANO

Inteligencia artificial (o casi)

Muchos expertos en los juegos de aventuras se preguntan con asombro cómo diablos hace el ordenador para entender las frases escritas por el jugador. Otros ni siquiera se lo preguntan, como si fuera obvio que el ordenador está dotado de una determinada facultad de comprensión. ¿Es o no es una máquina inteligente?

La respuesta es: NO, el ordenador no es una máquina inteligente. El ordenador es tonto como un asno, aunque, en cambio, es el soldado perfecto, el eterno sueño de los generales de todas las épocas: dotado de poderes notables, obedece ciegamente y sin emociones cualquier orden, sin preguntarse los motivos y sin prever las consecuencias. Lo hace todo al pie de la letra, sin comprender mínimamente lo que está haciendo y sin preguntarse si el significado literal de las instrucciones recibidas corresponde realmente al deseado por el programador (caso bastante raro).

La inteligencia no está en el ordenador, sino en el programa. Este último, es el momento de recordarlo, sólamente es una secuencia de instrucciones escritas por un programador más o menos despierto. La inteligencia que demuestra el ordenador es la de quien ha escrito el programa. La ventaja del ordenador es que, con un poco de técnica, es posible escribir un programa adecuado para resolver no sólo un problema específico, sino todos los problemas del mismo tipo, ahorrando así bastante trabajo. En el capítulo anterior hemos dado un ejemplo de esta posibilidad, mostrándoles un programa que "entiende" el concepto de dirección.

Si lo han leído bien, tendrán claro que el programa no entiende nada, sino que se limita a desplazar números y reproducir escritos según reglas bien definidas (la secuencia de instrucciones que forma el programa). La definición de estas reglas es, precisamente, tarea del programador que quiera darle a la máquina una apariencia de comportamiento inteligente.

El analizador sintáctico (parser)

Hay que escribir, por lo tanto, un programa capaz de simular la comprensión de los conceptos que están en la base de los juegos de aventuras y que, fundamentalmente, se refieren a

- plano, lugares y desplazamientos;
- objetos fijos y objetos transportables;
- comprensión y ejecución de las acciones del aventurero.

El primer problema (plano y desplazamientos) ya está resuelto. El segundo se puede afrontar fácilmente asociando a cada objeto una cadena (su descripción) y un número (el lugar en el que se encuentra). Hablaremos más adelante de los detalles. Decididamente, el problema más complejo es el tercero: entender la frase escrita en el teclado y modificar, en consecuencia, el estado del juego.

El programa que efectúa el análisis de una frase (o más generalmente, de una secuencia de símbolos) sobre la base de un conjunto de reglas (o sintaxis), es llamado parser o analizador sintáctico. Un juego de aventuras consiste, sustancialmente, en un parser que, sobre la base de las palabras introducidas y de sus asociaciones, decide el comportamiento sucesivo del programa. Para escribir este programa es necesario, primero, definir las reglas que gobiernan el comportamiento. Para hacer esto, intentaremos clasificar las frases posibles escritas por el jugador:

- una dirección (desplazamiento), por ejemplo "NORTE";
- una acción, expresada con una sola palabra, por ejemplo "INVENTARIO";
- una acción sobre un objeto, por ejemplo "APRIETA EL VERDE".

Por ahora no entraremos en más detalles, dejándolos para el capítulo dedicado a las acciones. Haremos notar sólo que una misma acción puede tener efectos distintos en lugares distintos (ejemplo: "Lee el cartel"), y que el objeto citado también puede estar ausente. Hay que tener en cuenta esta y otras posibilidades si se quiere realizar un juego con un buen nivel de simulación.

Análisis del léxico

Para facilitar el trabajo del analizador sintáctico es necesario separar las palabras que componen la frase (sin tomar en cuenta eventuales espacios). Pero la manipulación de las palabras como cadenas exige bastante espacio: es más práctico asignar un número (código) a cada vocablo y pasar al parser sólo los códigos de las palabras que componen la frase. También hay que descartar las palabras erróneas, por ejemplo "Dija".

Estos trabajos de menor importancia son tarea del analizador del léxico, la subrutina que recibe una cadena que contiene la frase introducida por el jugador, la separa en vocablos diferentes y restituye sus códigos numéricos. A veces, el analizador del léxico hace un trabajo que sería competencia del parser: reconoce y deja a un lado los artículos, dado que no alteran el significado de la frase y añadirían un trabajo inútil al parser. Por ejemplo, si el jugador ha escrito:

"REPARA LA ANTENA"

la subrutina de análisis del léxico separa las tres palabras "REPARA", "LA", y "ANTENA". El artículo "LA" se descarta y se restituyen los códigos numéricos correspondientes a "REPARA" (27) y "ANTENA" (69). Será tarea del parser decidir si esta pareja de códigos corresponde a una frase válida en el lugar y en la situación en que se encuentra el aventurero. Dado que nuestro parser es bastante sencillo, las palabras sucesivas a la segunda (excluidos los artículos) son, simplemente, ignoradas. Por lo tanto no son posibles frases como "Mira detrás del cartel" y similares.

Intérprete y datos

Estamos casi preparados para empezar; los detalles podemos examinarlos sobre la marcha. Desde el próximo capítulo se empieza a escribir la aventura. Resumimos cómo funciona la técnica empleada para "La nave espacial condenada":

- el jugador introduce una frase;
- el analizador del léxico la separa en palabras y asigna un código a cada una;
- el parser decide si los códigos (las palabras) indican una acción válida;
- si es así, se llama a la subrutina BASIC que ejecuta la acción correspondiente.

No hemos citado las direcciones, pues también éstas, efectivamente, constituyen una acción como todas las demás, y no hay ninguna necesidad de tratarlas diferentemente.

El programa final está compuesto por una colección de datos, por una parte variables (las acciones BASIC) y por otra parte fijos (analizador del léxico, parser y subrutinas en conexión). Esta parte fija es perfectamente reutilizable para otras aventuras, sin necesidad de modificaciones. Dado que interpreta una serie de datos proporcionados por el programador, hemos llamado "intérprete" a esta sección reutilizable del programa. También algunas de las subrutinas BASIC fundamentales (las usadas para las direcciones, "Save", "Coge", etc.) son reutilizables tal cual o con pocas modificaciones.

El programa se compone, por tanto, de tres partes:

- el intérprete, utilizable sin modificación para otras aventuras;
- los datos de la aventura (vocablos, plano, descripciones, etc.);
- las subrutinas BASIC de las acciones (algunas de ellas reutilizables).

Una buena noticia: para escribir una aventura no es necesario comprender el funcionamiento del intérprete, que constituye la parte decididamente más compleja del programa. De todas formas, para quienes estén interesados en la última parte del libro daremos una detallada descripción técnica de su funcionamiento.

CAPITULO VII

PLANO, DICCIONARIO Y OBJETOS

Nuestra aventura

El objeto de este libro es capacitarles para escribir su propio juego de aventuras. Refiriéndonos al programa de la nave espacial condenada, les recordamos que el intérprete, formado por las líneas que van hasta la 1160, puede ser reutilizado sin modificación (aparte del ajuste de algún GOSUB que llama a las subrutinas).

Desde este capítulo iniciamos la descripción de las diferentes fases de la construcción de una aventura, utilizando como ejemplo la presentada en el segundo capítulo. Al terminar, y para los que estén interesados, describiremos también el funcionamiento del intérprete.

El plano

Una vez establecida la disposición de los lugares en los que se desarrolla la aventura y sus conexiones relativas no queda más que introducir estos datos en el programa. Como hemos dicho en el capítulo anterior, se trata solamente de escribir una lista de instrucciones DATA.

El plano de los lugares en los que se desarrolla "La nave espacial condenada" está representado en la figura 1, y hemos añadido números que identifican cada lugar. Hay 11 lugares numerados del 1 al 11.

Para cada lugar es necesario proporcionar al programa una descripción del mismo y los números de los seis lugares colindantes (ver Capítulo 5). Con referencia al listado (reproducido en

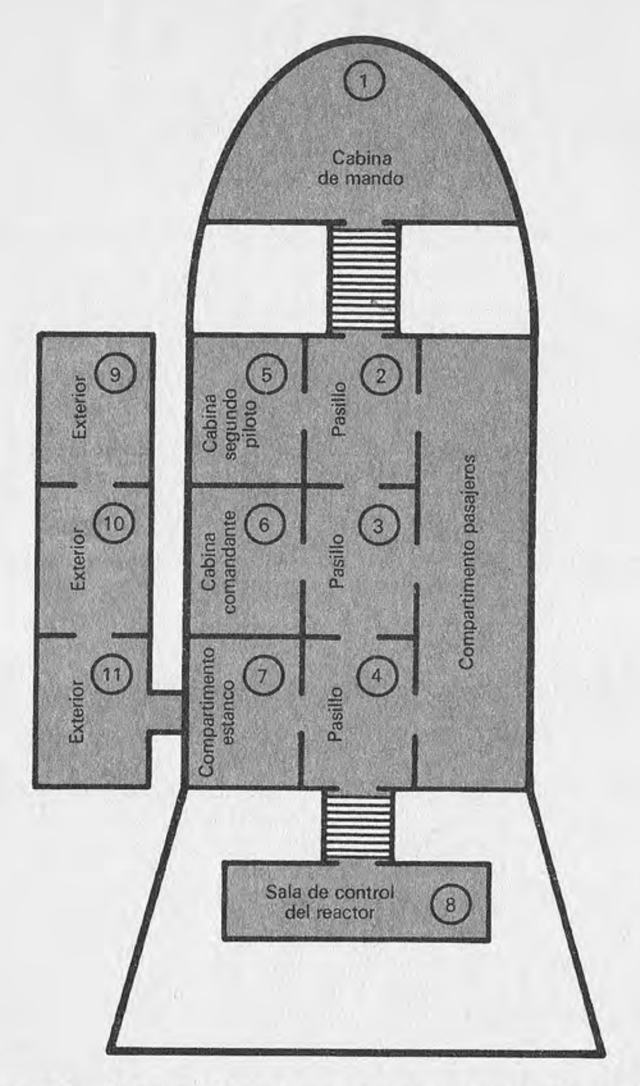


Figura 1.—Plano de la nave espacial de pasajeros "Neutronia".

el Capítulo 2), las líneas desde la 4130 a la 4230 contienen estas informaciones, en el orden correspondiente al número del lugar (el lugar 1 en la línea 4130, y así sucesivamente).

Nótese que las descripciones están incompletas: para ahorrar memoria hemos omitido el punto final y el "Estás" inicial, que es

añadido automáticamente por el intérprete en el momento de imprimir la descripción del lugar (por si quieres saberlo, línea 760).

Además, los números que representan los lugares adyacentes en las seis direcciones no están escritos separadamente, sino recogidos en una sola cadena de doce caracteres (también para ahorrar memoria). Los dos primeros caracteres indican el lugar adyacente al Norte, los dos sucesivos el adyacente al Sur y, a continuación, Este, Oeste, Arriba y Abajo.

Dado que es necesario indicar siempre dos caracteres, los números de una sola cifra van precedidos de cero.

Al término de las líneas de DATA, el "FP" indica el final de los planos (línea 4230).

En resumen, para introducir un plano deben de:

- dibujar cuidadosamente el croquis, con todas las conexiones;
- numerar los lugares con números progresivos;
- para cada lugar escribir una descripción y los números de los lugares adyacentes en las seis direcciones, en el orden N,S,E,O,A,B. Por ejemplo:

  (Estás) en el pasillo, 2,4,0,6,0,0

  Recordemos que el cero indica una dirección no admitida. Después vuelvan a escribir lo seis números seguidos, extendiéndolos a dos cifras si es necesario:

  02 04 00 06 00 00

  Y, quitando los espacios, obtendrán la cadena:

  "020400060000"

  La descripción y esta cadena, separadas por una coma, forman la línea de DATA que utiliza el intérprete (confrontar con la línea 4150);
- por último, no olviden la cadena "FP" que le indica al intérprete el final del plano. Pueden escribirla en una línea aislada de DATA o, mejor, al final de los DATA del último lugar (línea 4230). Si lo olvidan obtendrán errores extraños al empezar el programa (quizá en otra línea, cuando el intérprete pierda el hilo al leer los DATA).

El diccionario

Una vez definido el plano es necesario escribir todos los vocablos que deberán ser "entendidos" por el intérprete. Cuanto más rico sea el vocabulario, más fácil será simular un comportamiento

"inteligente" del juego. "La nave espacial condenada" utiliza unos 50 vocablos.

El trabajo se simplifica mucho si se definen desde el principio el mayor número posible de vocablos, añadiendo luego sólo los referentes a variaciones o posibles mejoras del juego. Empezaremos, pues, por catalogar los términos necesarios, asociando a cada uno un número (a nuestra elección) que el intérprete utilizará en lugar del vocablo, con un gran ahorro de espacio.

Para una mejor organización es muy cómodo subdividir las palabras en cuatro grupos:

- Direcciones y órdenes generales, utilizables en otras aventuras (ej. "Norte", "Coge", "SAVE").
- Verbos (ej. "Abre", "Salta").
- Objetos transportables (ej. "Llave").
- Objetos fijos y de decoración (ej. "Armario").

La figura 2 nos muestra los vocablos usados en nuestra aventura espacial, con sus respectivos códigos. Hemos reservado los códigos inferiores a 20 para los términos de uso general (si no los cambian les resultará muy fácil volver a utilizarlos en otros juegos). Después hemos subdividido los restantes en tres categorías, haciendo partir los códigos desde números arbitrarios (20, 50 y 60). Los colores (códigos 62, 63 y 64) sirven para frases tipo "Pulsa el verde" o "Pulsa el rojo", y por lo tanto son tratados como objetos fijos, de forma análoga a "Pulsa la tecla". Los artículos (que se ignoran) aparecen todos bajo el mismo código (7). Los sinónimos tienen el mismo código y están escritos en la misma línea (ej. "O" y "Oeste").

Los códigos son siempre números arbitrarios comprendidos entre 1 y 98, sin ninguna implicación de orden o espacio ocupado. Una subdivisión como la que hemos mostrado siempre será cómoda para distinguir los distintos tipos de palabras.

Surge ahora un problema técnico: la búsqueda del diccionario es demasiado lenta. Cuando el analizador del léxico ha aislado una palabra (ej. "ANTENA") y llama a la subrutina que consulta el diccionario para encontrar su código (el número que acabamos de asignarle), esta subrutina tiene que recorrer todos los vocablos hasta que encuentra la palabra buscada. Si ésta está al final de la lista, o no existe, puede pasar un tiempo relativamente largo, que de todas formas es molesto para quien juega.

La mejor solución es tener el diccionario en orden alfabético, para así poder usar una subrutina binaria de búsqueda rápida (ver el n.º 9 de la B.B.I.). Dado que la ordenación requiere un cierto tiempo, es mejor insertar los vocablos ya ordenados por orden alfabético. Esto quiere decir que el tiempo ahorrado por el juga-

La nave espacial condenada: diccionario

Código	Palabra	Código	Palabra	Código	Palabra
1	N,NORTE	2	S,SUR	3	E,ESTE
4	O,OESTE,W	5	A,ARRIBA, SUBE	6	B,ABAJO,BAJA
7	EL,LO,LA	8	COGE	9	DEJA
10	MIRA	11	SAVE	12	LOAD
13	COSA, INVENTARIO				
20	VISTE,PONTE	21	QUITA	22	ABRE
23	TIRA	24	EMPUJA	25	LEE
26	APRIETA	27	REPARA, ARREGLA		
50	CASCO	51	MONO	52	TRAJE
53	SEGUNDO	54	LLAVE	55	MANUAL
60	CARTEL	61	PULSADOR, TECLA	62	VERDE
63	AMARILLO	64	ROJO	65	PALANCA
66	INDICADOR	67	ARMARIO	68	CAMA
69	ANTENA	70	ETIQUETA		

Figura 2.—Vocablos del diccionario en el orden correspondiente a sus códigos.

dor se paga con un poco más de trabajo por parte del programador (generalmente merece la pena, después de todo se hace sólo una vez). Por lo tanto, al trabajo: el resultado final se aprecia en la figura 3.

Naturalmente, ahora los sinónimos aparecen repartidos por diferentes puntos de la lista. Cuando ordenen el diccionario compruébenlo bien: una correcta ordenación alfabética es fundamental. Si no, algún vocablo podría no ser reconocido. ¡Si tienen dudas... consulten un diccionario!

Volviendo al programa, las líneas de DATA desde el 3990 a la 4090 contienen el diccionario, en orden alfabético. Cada palabra va seguida por su código. El "FD" final indica al intérprete el final del diccionario.

La nave espacial condenada: diccionario ordenado

Palabra	Código	Palabra	Código	Palabra	Código
A	5	EL	7	PALANCA	65
ABAJO	6	EMPUJA	24	PONTE	20
ABRE	22	ESTE	3	PULSADOR	61
AMARILLO	63	ETIQUETA	70	QUITA	21
ANTENA	69	INDICADOR	66	REPARA	27
APRIETA	26	INVENTARIO	13	ROJO	64
ARMARIO	67	LA	7	S	2
ARREGLA	27	LEE	25	SAVE	11
ARRIBA	5	LLAVE	54	SEGUNDO	53
B	6	LO	7	SUBE	5
BAJA	6	LOAD	12	SUR	2
CAMA	68	MANUAL	55	TECLA	61
CARTEL	60	MIRA	10	TIRA	23
CASCO	50	MONO	51	TRAJE	52
COGE	8	N	1	VERDE	62
COSA	13	NORTE	1	VISTE	20
DEJA	9	O	4	W	4
E	3	OESTE	4		


Figura 3.—Vocablos del diccionario en orden alfabético.

Resumiendo:

- catalogüen los vocablos subdividiéndolos en grupos;
- asignen un código a cada vocablo o grupo de sinónimos;
- ordenen los vocablos en orden alfabético asociando a cada uno su código;
- incluyan lo anterior en el programa, en forma de DATA.

Los objetos

El aventurero se encuentra con objetos de muchos tipos. Algunos de éstos se pueden coger y transportar (como una llave),

otros simplemente forman parte de la decoración (como una cama). Todos pueden ser aludidos o manipulados (por lo menos con "Mira"). Los objetos transportables pueden encontrarse en varios sitios, o en posesión del aventurero o, provisionalmente, "fuera del juego".

El intérprete necesita una lista de los objetos con sus distintas características. Los objetos se numeran, asociando a cada uno de ellos un número a partir de 1. Este número no tiene nada que ver con el código asociado en el diccionario con el nombre del objeto, pero sirve para que el intérprete (y el programador) se refieran al objeto en cuestión. El orden es arbitrario, pero hay que tener en cuenta que los objetos son descritos por el jugador (veo...) por orden de número.

A cada objeto se le asocia una cadena (su descripción) y otros dos números: el código asociado en el diccionario a la palabra que lo describe y el lugar en el que se encuentra. La Figura 4 muestra la lista de objetos usados en "La nave espacial condenada".

El lugar contiene también otras informaciones:

- si el lugar es cero, el objeto es transportado por el aventurero;
- si el lugar es negativo, el objeto no se puede coger (ej. la cama). Este es un típico truco para tener dos informaciones distintas (lugar y posibilidad de cogerlo) en el mismo número;
- si el lugar es -99, el objeto está en el "limbo", o sea, fuera del juego.

Por ejemplo, el objeto número 6 está descrito como un cartel en el diccionario tiene el código 60 ("CARTEL"), se encuentra en el lugar 4 (final pasillo Sur) y no se puede coger (el número del lugar es negativo).

Puede haber varios objetos con el mismo nombre (y por lo tanto con el mismo código de diccionario), pero ninguno podrá coger (ej. las camas 7 y 9, y las teclas 2, 12, 13, 16, 17, 19) de forma que no puedan encontrarse nunca en el mismo lugar (el programa no se confunde a menudo, pero el jugador sí).

En la lista de DATA los objetos están catalogados en la misma forma, en el orden de su número y acabando con "FO" (final de objetos). El número no está explicitado en la lista: el primer objeto catalogado tiene el número 1, el segundo el número 2, y así sucesivamente. Las líneas desde la 4400 a la 4510 contienen la lista de los objetos utilizados en la aventura (idéntica a la de la figura 4).

La nave espacial condenada: objetos

Núm.	Objeto	Código	Lugar
1	un indicador	66	−1
2	un pulsador	61	−1
3	una etiqueta	70	−1
4	un mono	51	1
5	un cartel blanco	60	−2
6	un cartel amarillo	60	−4
7	una cama	68	−5
8	un armario	67	−5
9	una cama	68	−6
10	un armario	67	−6
11	un casco	50	6
12	un pulsador rojo	61	−7
13	un pulsador verde	61	−7
14	un indicador	66	−8
15	una palanca	65	−8
16	un pulsador rojo	61	−8
17	un pulsador verde	61	−8
18	un cartel rojo	60	−8
19	un pulsador amarillo	61	−8
20	el segundo piloto	53	9
21	una antena parabólica	69	−9
22	un traje	52	−99
23	una llave	54	−99
24	un manual	55	−99


Figura 4.—Objetos existentes en la aventura.

Resumiendo:

- catalogar los objetos utilizados y asignar a cada uno un número progresivo;
- establecer la descripción de cada objeto (ej. una "antena parabólica");
- escribir el código de la palabra en el diccionario (ej. el código de Antena, o sea, 69);
- escribir el lugar en el cual se encuentra el objeto al comienzo del juego (ej. 9 por la antena), -99 si está fuera del juego o cero si está transportado. Poner el signo menos si el objeto no se puede coger (como en el caso de la antena, cuyo lugar por lo tanto es -9);
- introducir todo esto en líneas de DATA.

Es importante tener bien marcada la lista de los objetos con el correspondiente número correlativo. En efecto, el intérprete crea una matriz LO% (Lugar Objetos) que contiene el lugar de cada objeto, en el orden de su número. El símbolo "%" (tanto por ciento) indica que se trata de una matriz de números enteros; solamente sirve para ahorrar memoria y, en algunos BASIC, también tiempo en los cálculos. Esta matriz LO% contiene casi todo el estado de la aventura si se exceptúan la posición del aventurero (contenida en la variable LU) y algunas informaciones accesorias. Todos los otros datos son constantes (no cambian en el curso del juego). Actuando sobre LO% el programador puede mover los objetos o transformarlos. Por ejemplo, la instrucción

```
LO%(23)=7
```

lleva el objeto 23 (la llave) al lugar 7 (el compartimento estanco).

Dado que la descripción de los objetos visibles es automática, la instrucción citada hace "aparecer" la llave, que puede ser cogida, mirada, etc. Con la misma técnica se puede transformar un objeto, por ejemplo haciendo desaparecer una luz verde (poniéndola en el lugar -99) y aparecer una luz roja (poniéndola en el lugar actual, LU o mejor -LU, dado que se trata de un objeto que no se puede coger). Desde el punto de vista del jugador siempre es el mismo objeto, pero el programador trabaja, en realidad, con dos objetos diferentes.

La manipulación del lugar de los objetos mediante LO% es el instrumento principal a disposición del programador.

Otros ejemplos:

```
LO%(6)=0
```

deja el objeto 6 (el casco) en posesión del aventurero, mientras que:

```
LO%(24)=LU
```

pone el objeto 24 (el manual) en el lugar actual. Si estaba en otra

habitación, "aparece"; si estaba en posesión del personaje es "dejado" en el suelo. Finalmente, la instrucción:

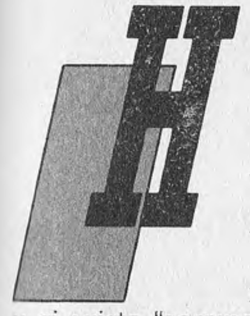
```
IF LO%(23)=0 THEN...
```

significa: si el objeto 23 (la llave) está en el lugar cero (transportado), o sea, si el aventurero posee la llave... sigue la consecuencia (por ejemplo, se puede abrir el armario).

CAPITULO VIII

¡ACCION!

Llevar a cabo las órdenes



Hemos llegado al punto más delicado de todo el mecanismo: la ejecución de las órdenes dadas desde el teclado. Es aquí donde se ve la "inteligencia" del programa (o la estupidez); veamos, por lo tanto, el problema en todos sus detalles. El jugador escribe una frase compuesta por una o dos palabras significativas (excluidos, por lo tanto, los artículos) que llamamos, respectivamente, "primera palabra" (o "verbo") y, si existe, "segunda palabra" (o "nombre"). El parser tiene, sobre la base de los códigos de estas palabras, que llamar a la subrutina que ejecuta la acción deseada.

Debería estar ya bastante claro lo que significa ejecutar una acción: se trata de imprimir un mensaje y/o modificar el estado de algunas variables, típicamente LU (el lugar del aventurero) y LO% (el lugar de uno o más objetos). Por ejemplo, la frase "Coge el manual" debe hacer partir una subrutina BASIC que controla si el manual está presente (y por lo tanto, se puede coger) y, en caso afirmativo, desplaza el manual mismo al lugar cero (transportado) e imprime el mensaje "Hecho".

Pero no es en absoluto conveniente escribir subrutinas BASIC diferentes para "Coge el manual", "Coge el mono", "Coge el traje", etc. Una sola subrutina puede ocuparse del verbo "Coge" en general.

Esta es la técnica utilizada en muchos juegos de aventuras, técnica que también está ilustrada en varios libros sobre el tema: el analizador sintáctico llama a la subrutina BASIC correspondiente a la primera palabra de la frase (el verbo), dejando a esta su-

brutina la tarea de distinguir entre los diferentes casos particulares. El "selector" de la subrutina a ejecutar funciona más o menos de esta forma:

```
500 IF C1=1 THEN GOSUB 1000
510 IF C1=2 THEN GOSUB 1010
520 IF C1=3 THEN GOSUB 1020
...
```

Si C1 es el código de la primera palabra de la frase, el funcionamiento está claro: la subrutina que comienza en la línea 1000 ejecuta las acciones referentes al verbo con código 1 (por ejemplo "Coge"), la 1010 las relativas al verbo con código 2, y así sucesivamente. Si los códigos parten de 1 y son correlativos, se utiliza un sistema mucho más simple y eficiente:

```
500 ON C1 GOSUB 1000,1010,1020,...
```

que tiene exactamente el mismo efecto. Si C1 vale 1 se ejecuta la subrutina número 1, o sea, la primera de la lista (equivale a un GOSUB 1000), si vale 2, la segunda, y así sucesivamente.

Problemas de comprensión

Como decíamos, el sistema ilustrado es el más utilizado. Probablemente el motivo reside en su sencillez: el parser no tiene que tomar decisiones, sino que deja todo el trabajo a las subrutinas que llama. Como ocurre a menudo, se trata de un planteamiento equivocado, aunque a primera vista parezca lo contrario: la sencillez del parser se paga con una mayor complicación de las subrutinas, que son muchas, mientras que el parser es uno solo, o bien, si así se elige, con un comportamiento "más tonto" del programa. Consideremos, por ejemplo, la frase "Coge el manual". Existen varias posibilidades:

- el manual está en el mismo lugar que el personaje,
- el manual se encuentra en otro lugar o fuera del juego,
- el manual está ya en posesión del aventurero,
- el manual no es un objeto que se pueda coger.

Si consideramos estas posibilidades en la subrutina de "Coge", deberemos hacer un trabajo análogo en muchas otras subrutinas ("Deja", "Mira", etc.). Si no las consideramos, la simulación y el realismo se pierden.

Habitualmente se utiliza una solución de este tipo: se considera que la segunda palabra es un objeto y se verifica que esté presente. Pero también esta "solución" presenta problemas. Consideremos la frase "Busca el manual": naturalmente el manual no está presente, por lo tanto el parser contesta "No entiendo" (mientras que desearía poder imprimir una frase apropiada, del tipo "Búscalo tú, yo no tengo ganas"). En este caso, el control de presencia es molesto. Una situación de este estilo se presenta, en "La nave espacial condenada", con los pulsadores de color: la frase "Coge el verde" se ejecuta aunque no exista en la habitación el objeto "Verde", sino sólo el objeto "Pulsador". Por otra parte, la acción "Lee el manual" se ejecuta sólo si el manual está presente.

En consecuencia, muchas aventuras comerciales entran ya en crisis con frases relativamente sencillas, como "Mira el árbol" que producen respuestas, por ejemplo, como "No noto nada de particular" aunque el árbol no esté presente, o, en otras situaciones similares, muchas contestan sistemáticamente "No entiendo".

Podríamos continuar con una lista de los problemas, de las soluciones parciales y de las correspondientes trampas y complicaciones, pero preferimos quedarnos en este punto e ilustrar una solución más flexible y que, a costa de una ligera complicación del parser (que lo hemos escrito nosotros de una vez por todas), ahorra trabajo y problemas a los futuros programadores (o sea, a ustedes).

Tabla de las acciones

El parser de "La nave espacial condenada" no es particularmente listo, aunque está por encima de la media de los contenidos en las aventuras comerciales y de los presentados en libros semejantes.

Nuestro parser funciona de esta forma: el programa consulta una tabla de acciones que contiene información sobre las acciones válidas, y en base a ella decide si ejecuta una subrutina BASIC, y cuál. Las subrutinas BASIC se numeran a partir de 1 y se las llama con una instrucción ON...GOSUB, como ya hemos visto anteriormente. La tabla puede reconocer cuatro tipos de frases, compuestas por:

1) Una pareja de palabras (verbo + nombre) en un determinado lugar (ejemplo "Aprieta el pulsador" en la sala de control del reactor).
2) Una palabra específica (verbo) seguida de otra cualquiera en un determinado lugar (ejemplo, "Mira XXX" en una habitación con ilusiones ópticas).

3) Una pareja de palabras (verbo + nombre) en un lugar cualquiera (ejemplo "Lee el manual", en cualquier sitio).
4) Una palabra específica (verbo) seguida de cualquier otra, en un lugar cualquiera (ejemplo, "Coge XXX", en cualquier sitio).

En todos estos casos es posible especificar si el objeto indicado por la segunda palabra debe estar presente para que la subrutina BASIC sea ejecutada, o si la acción es válida de todas formas también en ausencia del objeto.

Se reconocen también dos tipos de frases, compuestas por una sola palabra, que en la práctica son como los tipos 1 y 2, pero con la segunda palabra ausente:

5) Una palabra (verbo) en un determinado lugar (ejemplo, "Este" en el pasillo).
6) Una palabra (verbo) en un lugar cualquiera (ejemplo, "Inventario").

Una cosa muy importante: la búsqueda en la tabla tiene lugar en el orden en el que hemos catalogado las diferentes posibilidades. Por ejemplo: si el jugador ha escrito "ESTE", el parser busca si está prevista la palabra "Este" en el lugar actual del aventurero (caso cinco). Si la encuentra, ejecuta la acción indicada. Si no la encuentra, mira si está prevista la palabra "Este" como válida en cualquier sitio (caso seis). La encuentra y llama a la subrutina indicada (la de desplazamiento). Si no la encontrase, imprimiría "No entiendo". Lo mismo ocurre en el caso de dos palabras: el parser mira primero si está previsto el caso uno (ese verbo y ese nombre en ese lugar), y solamente si no lo encuentra pasa a los casos 2,3,4, y finalmente imprime "No entiendo". La regla es que el caso particular tiene preferencia sobre el general. En otros términos: habitualmente la palabra "Este" tiene un determinado efecto (mueve hacia el este), pero en un determinado lugar específico queremos que llame a una subrutina en particular. Si no está todo claro no se preocupe (todavía). Vamos a ver algún ejemplo que disipe las dudas que han podido ir surgiendo.

Dirección: Norte

Comencemos con el caso más sencillo: las direcciones. Hemos dicho anteriormente que las direcciones no requieren un tratamiento especial: son palabras como las otras, se trata solamente de que ejecuten la subrutina adecuada. Esta subrutina será una sola para todas las direcciones y decidirá su actuación en base

al código de la primera (y única) palabra, cuyo código se encuentra contenido en la variable C1 (con un poco de imaginación adivinarán que C2 contiene el código de la segunda palabra).

Por lo tanto, la tabla de las acciones tiene que contener estas informaciones:

- si la frase contiene sólo la palabra "N" o un sinónimo (código 1), ejecuta la subrutina BASIC número 1 (que se ocupa de las direcciones);
- si la frase contiene sólo la palabra "S" o un sinónimo (código 2), ejecuta la subrutina BASIC número 1 (siempre la misma);
- si la frase contiene sólo la palabra "E" o un sinónimo (código 3), ejecuta la subrutina BASIC número 1 (ídem, como arriba).

Y así sucesivamente para las otras tres direcciones. Por lo tanto habrá seis informaciones en la tabla de las acciones. Cada información consiste en dos números: la síntesis de la frase y el número de la subrutina a ejecutar.

La síntesis de la frase es un número que se calcula de la manera siguiente:

```
LU*10000+C1*100+C2
```

Dicho más claramente: las primeras dos cifras indican el lugar en el que tiene efecto la frase (cero si vale cualquier sitio), las dos siguientes son el código de la primera palabra y las dos últimas son el código de la segunda palabra (cero si falta). Por lo tanto, la síntesis de la frase "NORTE", válida en cualquier sitio, se construye así:

Lugar:	00(cualquier sitio, se puede omitir)
Código1:	01(código de "NORTE" o "N")
Código2:	00(ninguna palabra).

El resultado o síntesis de la frase (obtenido adosando los tres datos) es 000100. Dado que se trata de un número y no de una cadena, es igual a 100. En la tabla de las acciones, que el programa lee al comienzo de las líneas de DATA, es inútil desperdiciar espacio con ceros no significativos; por eso, para que se ejecute la subrutina número 1 en respuesta a la frase "NORTE", basta con escribir:

```
DATA 100,1
```

Del mismo modo, la síntesis de "SUR" es 200, la de "ESTE" 300, la de "OESTE" 400, la de "ARRIBA" 500 y la de "ABAJO" 600. Dado que todas estas frases deben llamar a la subrutina número 1, la tabla de las acciones contendrá seis informaciones, cada una compuesta de síntesis de frase y número de la subrutina a ejecutar:

DATA 100,1,200,1,300,1,400,1,500,1,600,1

Miremos la línea 4270 del programa: comienza exactamente con estos datos. Para acelerar la búsqueda de las acciones, la tabla se escribe en el orden correspondiente al número de síntesis de frase. Por ahora no hay problemas, pero es importante no olvidarlo.

Por lo tanto, si el jugador escribe la frase "NORTE" en el lugar 14, el proceso siguiente será:

- el analizador del léxico devuelve los códigos de las dos palabras: C1 = 1 (código de "Norte") y C2 = 0 (código de no hay palabra);
- el parser compone la síntesis de LU,C1 y C2, o sea, el número del lugar actual, código de la primera palabra y código de la segunda palabra. Obtiene 140100 y busca este número en la tabla de las acciones. Supongamos que no lo encuentra: querría decir que la acción "Norte" no está prevista en el lugar 14;
- el parser compone entonces la síntesis de 01 y 02 solamente, o sea 100, y la busca en la tabla de las acciones. La encuentra: es una acción prevista como válida en todos los lugares. Lee el número de la acción a ejecutar: 1;
- el parser llama a la acción número 1 utilizando el selector de las acciones (línea 1710), que cede el control a la subrutina de la línea 1780, o sea, a la subrutina BASIC de dirección (¡finalmente!).

Para no dejar misterios sin solucionar, vamos a ver lo que ocurre en la mal afamada subrutina que ejecuta los desplazamientos (líneas 1780-1800). Ocurre exactamente lo que ya les hemos mostrado en el capítulo 5, con la única diferencia de que el número del nuevo lugar no se lee por una matriz numérica sino por una matriz de cadena. DI$(LU) no es otra cosa más que la cadena de doce cifras que se ha introducido a continuación de la descripción del lugar número LU (ver capítulo siete, plano, y líneas 4130-4230).

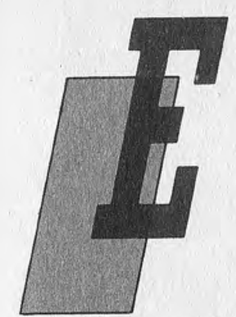
El MID$ de la línea 1780 aísla las dos cifras correspondientes a la dirección elegida, basándose en el código C1 (las primeras dos para "Norte", las dos siguientes para "Sur", etc.), y el VAL de

la misma línea las convierte finalmente en un número; el número del nuevo lugar que, provisionalmente, está en A. Si la dirección no está permitida, la línea 1790 lo hace notar; de otro modo, la línea 1800 ejecuta el desplazamiento. Destaquemos que la línea 1780 sólo funciona si los códigos del diccionario de las seis direcciones son 1,2,3,4,5,6. ¡Por lo tanto, no los cambien!

CAPITULO IX

ALGUNAS ACCIONES

Las tres más sencillas



En la figura 1 pueden ver nuestros apuntes relativos a las acciones generales de "La nave espacial condenada". Hemos pensado proponérselos sin retocarlos, aunque contengan alguna imprecisión y no siempre sean perfectamente coherentes, porque nos parecían adecuados para mostrar el trabajo paciente que hay entre bastidores de una aventura, aunque sea tan sencilla como ésta.

Los apuntes están organizados de esta manera: en la primera columna se han escrito las frases que tiene que reconocer el programa (reagrupadas si tienen que ejecutar la misma subrutina, como las seis direcciones), en la segunda columna está la síntesis numérica de la frase (número lugar, código primera palabra, código segunda palabra), sigue el número de la acción a ejecutar (subrutina BASIC) y, finalmente, hay una descripción concisa de la acción misma. Las acciones de la figura 1 son las generales, o sea, las válidas en todos los lugares e independientes de la aventura específica (salvo algunas excepciones, que ya veremos).

Comencemos con tres acciones sencillas: "SAVE", "LOAD" e "INVENTARIO" que, respectivamente, se ocupan de registrar la situación actual (muy aconsejada antes de emprender acciones imprudentes), retomar una situación anterior y cataloga los objetos que posee el aventurero (con el cómodo sinónimo "COSA").

El mecanismo de ejecución de estas acciones funciona exactamente como el de las direcciones, haciendo ejecutar la adecuada subrutina BASIC de acción (respectivamente 5, 6 y 7). Les re-

La nave espacial condenada: acciones generales

Frase	Síntesis	Subrutina	Acción
N	0100		
S	0200		
E	0300		
O	0400		
A	0500		
B	0600	1	Mueve
COGE XXX	0899	−2	si Lugar(OB) = 0 "Ya hecho" si OB = no se puede coger "No es posible" si OB = Mono y lugar (Traje) = 0 "Quítate antes el Traje" si OB = Traje y lugar(Mono) = 0 "Quítate primero el mono" si no lugar(OB) = 0 si OB = Casco,Mono o Traje "Ahora lo Tienes puesto" si no "Hecho"
DEJA XXX	0999	3	si OB = 0 o Lugar(OB)< >0 "No lo Tienes" si LU> = 9 {fuera} lugar(OB) = −99 "Perdido en el espacio" si no lugar(OB) = LU "Hecho"
MIRA XXX	1099	−4	"Nada de particular"
SAVE	1100	5	guarda situación
LOAD	1200	6	recobra situación
COSA,INVENTARIO	1300	7	imprime inventario

Figura 1.—Primeros apuntes de las acciones generales de "La nave espacial condenada".

cordamos que los códigos de las palabras se encuentran en el diccionario preparado con anterioridad (capítulo 7).

Las subrutinas BASIC de "Save" y "Load" se encuentran en las líneas 2000 y 2060 y, naturalmente, dependen del ordenador utilizado. Son muy primitivas, en el sentido de que sólo está permitido el rescate de una única situación, con nombre fijo ("ASTRO", definido en la línea 1010).

La subrutina de "Inventario" (línea 2120) hace uso, en cambio, de una subrutina del intérprete, la que describe los objetos presentes en un determinado lugar. Dado que los objetos poseídos por el aventurero están en el lugar cero, vale reutilizar la misma

subrutina. Conviene dejar tal cual esta acción. Como inciso, el GOTO 590 al final de la línea 2120 es, conceptualmente, un GOSUB 590 seguido de un RETURN. El efecto es idéntico y se ahorra tiempo (en la práctica, la subrutina de inventario retorna utilizando el RETURN que está al final de la subrutina llamada).

Mira

La acción "MIRA XXX", en cambio, es más complicada: debe ejecutarse como respuesta a una frase compuesta por la palabra "MIRA" seguida de una palabra válida cualquiera y sirve en un lugar cualquiera. La síntesis de la frase se construye como antes:

Lugar:	00(cualquier sitio, se puede omitir)
Codigo1:	10(código de "MIRA")
Codigo2:	99(cualquier palabra válida).

Una novedad: indicando 99 como código de la segunda palabra se entiende "cualquier palabra válida, siempre que exista". La síntesis de la frase es, por lo tanto, 1099 y, dado que tenemos que ejecutar la acción 4, en la tabla deberemos escribir:

```
DATA 1099,4
```

En cambio, en la tabla se encuentra:

```
DATA 1099,-4
```

(más o menos hacia la mitad de la línea 4290). Esta es una comodidad más: si el número de la subrutina es negativo, significa que el parser tiene que controlar que la segunda palabra se refiera a un objeto presente o transportado, es decir, situado en el lugar actual o en posesión del aventurero. En caso contrario, imprime "Aquí no está» y se niega a llamar a la subrutina indicada.

Dado que para mirar un objeto este último tiene que estar presente, ponemos el signo menos y nos aseguramos así que la subrutina número 4 sea ejecutada sólo si el jugador mira a un objeto realmente visible.

Como pueden ver en la línea 1970, la subrutina número 4 se limita a imprimir "No noto nada de particular". En efecto, se trata de la subrutina general de "Mira...", que está al final de la lista (caso 4 de los enumerados en el capítulo 8) y sirve para dar, de todas formas, una respuesta, en ausencia de acciones más interesantes (como "mira el indicador"). En jerga técnica es una subrutina de

"default", o sea, a ejecutar en ausencia de otras subrutinas con preferencia más alta.

Coge

Llegamos ahora a dos acciones fundamentales en los juegos de aventuras: coger y dejar los objetos. Comencemos por la primera.

Si se quiere un mínimo de realismo, en la acción "COGE" es necesario considerar que el objeto citado:

- tiene que estar presente;
- no puede estar ya en posesión del aventurero;
- tiene que poder cogerse.

La primera condición es automática: basta poner un signo menos delante del número de la acción (ver la figura 1: el número de la acción a ejecutar es -2). Para satisfacer las otras dos, la subrutina llamada se plantea más o menos de esta manera:

- si el objeto ya está en poder del aventurero imprimir "Ya hecho" y volver;
- si el objeto es del tipo de los que no se pueden coger (lugar negativo, ver capítulo 7), imprimir "No es posible" y volver.
- si todo es correcto, desplazar el objeto al lugar cero (transportado), imprimir "Hecho" y volver.

Para referirse al objeto citado en la frase, se utiliza la variable OB, en la cual el intérprete pone el número del objeto (¡no el código del diccionario!; ver capítulo 7) citado como segunda palabra.

Atención: si el objeto no está presente en el lugar LU o transportado, OB contiene cero y no se utiliza. Por lo tanto, LO%(OB) es el lugar del objeto citado como segunda palabra. Nosotros tenemos la costumbre de llevar nuestros apuntes en una forma de "pseudocódigo", o sea, en una especie de lenguaje estructurado del cual les doy, en seguida, un ejemplo reescribiendo parte de la subrutina de "Coge":

```
si lugar(OB)=0
    "Ya hecho"
si OB=no se puede coger
    "No es posible"
Si no lugar(OB)=0
    "Hecho".
```


No hay mucha diferencia respecto a lo anterior; es sólo más conciso. Las palabras "si" y "si no" o "de otro modo" se pueden sustituir por las equivalentes en inglés "if" y "else"; la ventaja está en la brevedad y en la semejanza con instrucciones BASIC, lo que simplifica el trabajo de traducción al programa. Además, escribimos simplemente entre comillas un resumen de los mensajes a imprimir y hacemos uso de la indentación, que consiste en desplazar hacia la derecha (indentar) las opeaciones (instrucciones) relativas a un determinado if, para verlas más claramente. Dado que este libro no es un curso de programación estructurada, nos pararemos aquí. De todas formas, pensamos que nuestros apuntes son bastante legibles.

Volviendo a la subrutina de "Coge", se puede traducir al BASIC de esta manera:

```
1830 IF LO%(OB)=0 THEN PRINT "YA HECHO":RETURN
1840 IF LO%(OB)<0 THEN PRINT "NO ES POSIBLE":RETURN
1870 LO%(OB)=0
1890 PRINT "HECHO":RETURN
```

Nótese cómo la subrutina corresponde exactamente a los apuntes. En particular, la línea 1840 verifica que el objeto citado se pueda coger comprobando que su lugar no sea negativo (si es negativo, significa "que no se puede coger"; ver capítulo 7).

La subrutina de "Coge" utilizada en "La nave espacial condenada" es ligeramente más complicada, porque tiene en cuenta algunos casos particulares: impide ponerse el mono si ya se tiene puesto el traje (y viceversa), e imprime "Ahora lo tienes puesto" en vez de "Hecho" si el personaje coge el casco, el mono o el traje. Verifiquen estas diferencias entre los apuntes de la figura 1 y las líneas 1830-1890 (en la práctica, se han añadido las líneas 1850, 1860 y 1880). En una de sus propias aventuras, basta con que quiten estas líneas añadidas para tener una subrutina general del "Coge".

Deja

Pasamos a la acción recíproca: la de dejar un objeto. Aquí el trabajo es más sencillo: si el objeto no es transportado no se puede dejar. Dado que queremos imprimir "No lo tienes" si el objeto no es transportado (esté presente o no), hay que evitar el control preliminar del intérprete (que imprimiría "Aquí no está" en caso de ausencia). Para evitar el control basta utilizar un número de acción positivo (efectivamente, es 3, y no -3).

Pero hay una trampa: Llegados a este punto no basta con mirar LO%(OB) para saber dónde está el objeto. Efectivamente, como hemos dicho antes, si el objeto no está presente o transportado OB vale cero y LO%(OB) producirá, teóricamente, un error. En realidad, LO%(O) contiene cero (este elemento no se ha utilizado nunca), pero esto no significa en absoluto que el objeto esté transportado, ¡significa sólo que estamos mirando una variable sin significado!; en suma, hay que comprobar que OB no valga cero, porque si no quiere decir que el objeto, ciertamente, no se puede dejar. Por lo tanto, la acción de "Deja" es:

```
si OB=0 o lugar (OB)<>0
     "No lo tienes"
si no
lugar (OB)=LU
     "Hecho"
```

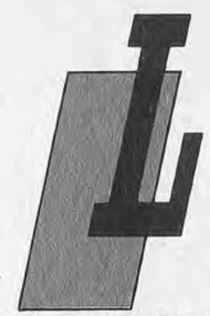
Así como para la acción "Coge", también para "Deja" hay un caso particular que concierne a "La nave espacial condenada": si el número del lugar es mayor o igual a 9 (o sea, 9, 10, 11) nos encontramos en el espacio exterior y cualquier objeto dejado se perderá, trasladándolo al lugar -99 (el "limbo"). Las líneas de programa desde la 1920 hasta la 1940 son la subrutina de "Deja".

Como en todos los programas no existe una única forma de resolver los problemas. Se hubiera podido escribir una subrutina más "limpia" para "Deja" y prever la frase "Deja XXX" en los lugares 9, 10, 11. Nuestra solución es más corta, pero quizá menos aconsejable para quien prefiere las cosas bien ordenadas. Esto mismo vale para "COGE EL CASO", "COGE EL MONO", y "COGE EL TRAJE", en donde, al querer señalar las tres acciones espaciales, las cosas resultaban más complejas. Era necesario, en la práctica, repetir cuatro veces los controles de la subrutina de "Coge". Probablemente no valía la pena.

CAPITULO X

A BORDO DEL "NEUTRONIA"

Acciones a medida



Las acciones que hemos visto hasta ahora son de carácter general, reutilizables para otras aventuras. Ahora se trata de escribir acciones específicas para una determinada aventura, precisamente la de "La nave espacial condenada". Ya deberían tener muy claro que, a diferencia de lo que ocurre en muchos otros programas para aventuras, estas acciones específicas funcionan exactamente igual que aquellas generales, gracias a la tabla de las acciones. También la subdivisión que hemos hecho en los apuntes es sencillamente una cuestión de comodidad.

No les describiremos detalladamente todas las acciones; nos limitaremos a comentar las más significativas, dejándoles el útil ejercicio de examinar las otras y entender cómo funcionan. Empezamos en seguida con las acciones válidas en todos los lugares de la nave espacial, que pueden ver resumidas en la figura 1.

Casco, Mono y Traje

Ya las tres primeras acciones ("Ponte Casco", "Ponte Mono" y "Ponte Traje", donde omitimos los artículos como hace el programa) merecen nuestra atención. Para simplificar el juego hemos establecido que coger uno de estos tres objetos equivale a ponérselo (véase la subrutina de "Coge" del capítulo anterior). Entonces parecería conveniente indicar simplemente "Ponte" como sinónimo de "Coge" (mismo código en el diccionario); de este modo,

La nave espacial condenada: acciones válidas en cualquier lugar

Frase	Síntesis	Subrutina	Acción
Ponte Casco	2050	−2	goto Coje XXX
Ponte Mono	2051	−2	goto Coge XXX
Ponte Traje	2052	−2	goto Coge XXX
Deja Casco Deja Mono Quita Casco Quita Mono	0950 0951 2150 2151	−10	if lugar(OB)<>0 "No lo tienes" if LU>=9 or (LU=7 & var2=1) {fuera} "Aaaagh" Muerto else goto Deja XXX
Quita Traje	2152	−3	goto Deja XXX
Mira Mono	1051	−11	"Es tu mono para activ. extravehic."
Mira Segundo	1053	−12	"Tiene el casco deteriorado"
Mira Cartel Mira Manual	1060 1055	−13	"¿No es mejor leerlo?
Mira Traje	1052	−14	"Pesado, tratado con plomo"
Lee Manual Abre Manual	2555 2255	−15	if lugar(OB)=LU "Cógelo con la mano antes" else "Instrucciones: para activar..."
Repara antena	2769	−29	"Necesitamos al segundo"

Figura 1.—Acciones de "La nave espacial condenada" válidas en cualquier lugar.

las frases "Coge Mono" y "Ponte Mono" tendrían exactamente el mismo efecto. Lamentablemente, no siempre la sencillez es buena: con este sistema se aceptarían frases como "Ponte Manual" o "Ponte llave". Con respecto a esto hay dos puntos de vista: el primero (típico de muchos autores) es el de "¿A quién le importa? Nadie dirá nunca una frase así". En cambio, nosotros pensamos que el realismo y el interés de una aventura se miden también, y mucho, por el cuidado en los detalles. Por suerte, el parser y la tabla de acciones hacen sencilla la adición de nuevas posibilidades, y así también los vagos podrán construir una aventura rica en detalles sin mucho esfuerzo.

Por ejemplo, es muy fácil hacer que "Ponte Casco" tenga el

mismo efecto que "Coge Casco". Es suficiente con poner en la tabla de acciones las síntesis de "Ponte Casco":

Lugar: 00(en cualquier sitio)
Ponte: 20
Casco: 50

Por lo tanto, la síntesis es 2050, número que se debe insertar en su lugar en la tabla (que, recordamos, va en orden de número de síntesis). A continuación se pone el número de la acción (subrutina BASIC) a ejecutar. Dado que la frase tiene que tener el mismo efecto que "Coge Casco", basta con poner el mismo número que la subrutina de "Coge", o sea, -2 (el signo menos, como siempre, quiere decir que el objeto tiene que estar presente). En la figura 1 la hemos indicado como "goto Coge XXX" para recordar que esta acción tiene el mismo efecto que la otra. Esto mismo sirve, obviamente, para "Ponte Mono" y "Ponte Traje".

Ahora, por tanto, la frase "Ponte Casco" tiene el mismo efecto que "Coge Casco", pero la frase "Ponte Llave" no es reconocida en ninguna parte y produce así la respuesta "No entiendo". En una aventura bien hecha también se debería de prever una respuesta estándar por defecto para el caso "Ponte XXX", o sea, para todos los intentos de ponerse objetos que no son adecuados (por ejemplo: "No sabía que tú pudieses cambiar de forma. ¿Puedes transformarte también en un gusano?"). Mejor aún, se debería de prever una respuesta adecuada para cada verbo cuando no sea usado en el modo o con el objeto correcto. Si no, la frase "No entiendo" en seguida se hará pesada: la variedad mantiene el interés del jugador. Con referencia a Casco y Mono observen (en la figura 1) las frases "Deja Casco", "Quita Casco", etc.

Dado que sería inverosímil permitir que un astronauta se quite el casco o el mono, sin sufrir daños mientras se encuentra en el vacío, hay que considerar esta posibilidad: si el personaje realiza la acción en el exterior de la nave espacial (lugar>=9), o en el compartimento estanco (lugar 7) cuando éste está abierto al vacío (variable 2, como veremos más tarde), sucede lo inevitable. Si no es ésa la situación se ejecuta la subrutina normal "Deja XXX". Para hacerlo es suficiente con ejecutar un GOTO en BASIC a la subrutina en cuestión, como se puede ver en la línea 2220 del programa. Las líneas 2210-2230 constituyen la subrutina de acción.

Acciones particulares y generales

Aunque en esta aventura no hay ejemplos adecuados, les recordamos que una misma acción puede tener un efecto distinto

en diferentes lugares. Por ejemplo, la frase "Pide ayuda" sólo funcionaría cerca de la jaula de los leones (imaginen ustedes el efecto), mientras que en otros sitios se contestaría con un desolador "Nadie está cerca para oírte".

Es cierto que en este caso se puede escribir una subrutina válida para todos los lugares (cuya síntesis empezará, por lo tanto, con 00) y efectuar una comparación (IF LU = 47 THEN...) al principio de la subrutina para establecer si nos encontramos en un lugar especial, pero es mejor utilizar dos subrutinas distintas: una general (válida en todos los lugares), que da una respuesta por defecto, y una específica (válida en un solo lugar) que realiza el efecto deseado. Es el parser el que se ocupa en dar preferencia a esta última si el aventurero se encuentra en el lugar indicado. Les recordamos que el orden de preferencia de las distintas acciones ya fue descrito en el capítulo 8.

Acciones específicas

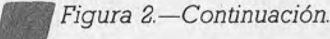
La figura 2 nos muestra la lista de las acciones que sólo tienen efecto si la frase se pronuncia en un determinado lugar. Si la frase se enuncia en lugar distinto al indicado, el parser no la encontrará; entonces mirará si la tabla contiene la misma acción válida en todos los lugares. Si no aparece imprimirá "No entiendo".

La nave espacial condenada: acciones válidas en un lugar específico

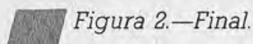
Cabina (1):			
Mira Indicador	011066	16	"Temperatura"
Lee etiqueta	012570	17	"SOS Galáctico, sólo para emergencias"
Aprieta pulsador	012661	18	"Antena exterior defectuosa"
Pasillo Norte (2):			
Lee Cartel	022560	19	"Escalera: entrada reservada..."
A	020500	20	LU=1 {cabina} if lugar(Segundo)=9 {fuera} "Si estuviera aquí..."
E	020300	21	"Mejor no despertar a los pasajeros"

Figura 2.—Acciones de "La nave espacial condenada" válidas sólo en un lugar.

Pasillo (3):			
E	030300	21	"Mejor no despertar a los pasajeros"
Pasillo Sur (4):			
Lee Cartel	042560	22	"O: Atención: hab. despresurizada..."
E	040300	21	"Mejor no despertar a los pasajeros"
Cabina Segundo (5):			
Abre Armario	052267	23	if lugar(Manual)< >−99 "Ya hecho" else if lugar(Llave)< >0 "Cerrado con llave" else "Hecho" Lugar(Manual)=LU Lugar(Traje)=LU
Cabina Comandante (6):			
Abre armario	062267	24	"Vacío"
Compartimento estanco (7):			
E	070300	37	if var2=0 {cerrado} LU=4 {pasillo} else goto 1 (Direcciones)
O	070400	38	if var 2=1 {abierto} LU=11 {exterior} LU=11 {exterior} else goto 1 (Direcciones)
Aprieta Pulsador	072661	25	"Aprieta Rojo o Aprieta Verde"
Aprieta Rojo	072664	26	if var2=1 {abierto} "Click" else "Si abre..." var2=1 if lugar(Mono)< > 0 lugar(Casco)< >0 "Aaaagh" Muerto if objetos móviles en LU lugar(objetos)=−99 "Perdidos en el espacio"
Aprieta Verde	072662	27	if var2=0 {cerrado} "Click" else "Se cierra..." var2=0 if lugar(Segundo)=0 and lugar(Llave)=−99 "Revive" lugar(Llave)=LU


Figura 2.—Continuación.

Sala control reactor (8):			
Lee Cartel Mira Indicador	082560 081066	28 16	"Emergencia al acecho: Traeros..." "Temperatura"
Aprieta Pulsa	082661	30	"Aprieta Rojo, Amarillo o Verde"
Aprieta Rojo	082664	31	"Click" "Tubería que pierde" if var1 = 0 var1 = 1 else var1 = 0 timer1 = timer1 div 2
Aprieta Amarillo	082663	32	"Click" if var1 = 1 var1 = 2 if lugar(Traje) < > 0 "Te sientes regular" else var1 = 0 timer1 = timer1 div 2
Aprieta Verde	082662	33	"Click" if var1 = 2 var1 = 3 if lugar(Traje) < > 0 "Exceso de radiaciones" Muerto else var1 = 0 timer1 = timer1 div 2
Empuja palanca	082465	34	"Clank" if var1 = 3 Final. ¡Felicidades! else var1 = 0 timer1 = timer1 div 2
Tira palanca	082365	35	"Clunk" var1 = 0 timer1 = timer1 div 2
Exterior (9):			
Mira Antena	091069	36	"Parece averiada"


Figura 2.—Final.

Nótese que los números de las acciones (subrutinas BASIC) a ejecutar son todos positivos: dado que en ninguna de estas acciones la segunda palabra indica un objeto móvil (podría suceder en otra aventura) y dado que el lugar es conocido (la síntesis de la frase empieza con el número del lugar en el que tiene que tener efecto la acción), no hay ninguna necesidad de comprobar la presencia del objeto citado (por ejemplo, el armario).

La frase "Repara Antena" se ha puesto entre las que son válidas en cualquier lugar (fig. 1), porque tiene que contestarse "Aquí no está" automáticamente, por efecto del signo menos, haciendo intuir así que la antena tiene que estar en alguna parte (muchos jugadores escriben esta frase en la cabina de mando).

La primera frase interesante es "A" ("Arriba") en el lugar 2 (pasillo). Su síntesis está hecha de la forma habitual, pero indicando un número de lugar:

Lugar:	02
Primera palabra:	05
Segunda palabra:	00

Por lo tanto, la síntesis es 020500, o sea 20500. Dado que una acción local tiene preferencia sobe una general, cuando el jugador escribe "A" en el lugar 2 no se ejecuta la subrutina general de movimiento (subrutina 1), sino la específica señalada en este caso (subrutina 20). Recordamos una vez más que las acciones de la tabla están en el orden del número de síntesis.

La subrutina que ejecuta la acción (2800-2830) no sólo se encarga de trasladar al aventurero a la cabina de mando (LU = 1), sino que imprime también un mensaje si el segundo piloto todavía no ha sido salvado. Esta es la técnica normal para hacer que suceda algo durante el traslado desde un lugar a otro.

Una subrutina parecida es la número 21, que imprime un mensaje ("Mejor no despertar a los pasajeros..."), pero no permite el desplazamiento (simplemente, no cambia LU).

En la acción "Abre armario", en el lugar 5 (cabina del segundo) se comprueba que el manual no haya aparecido antes. Sólo si aún está en el "limbo" (lugar -99), se comprueba si el personaje posee la llave. Y únicamente en este caso aparecen el traje y el manual. La subrutina BASIC está en las líneas 2960-2980; a partir de ahora no les indicaremos más líneas. Pueden buscarlas ustedes leyendo las REM, o bien las instrucciones ON... GOTO...

El compartimento estanco

El compartimento estanco es un lugar interesante, aunque no ponga muchos problemas a los jugadores amantes de la ciencia ficción. Dado que puede estar en dos estados, es decir, abierto hacia el pasillo o hacia el espacio exterior, es necesaria una variable para saber en qué estado se encuentra. La variable, que hemos llamado var2 en las notas, en realidad es la variable BASIC V2 (si hay una V1, ¿cómo se han enterado?). Si vale cero el comparti-

mento estanco está cerrado (con respecto al espacio exterior), si vale 1 está abierto. El pulsador rojo lo abre y el verde lo cierra.

Sería absolutamente erróneo, y no sólo inútil, indicar un número de subrutina negativo para las acciones "Aprieta rojo" y "Aprieta verde". En efecto, el parser buscaría los códigos correspondientes en el diccionario a las palabras "Rojo" y "Verde", que no existen; existen sólo dos objetos con código "Pulsador" y son precisamente los números 12 y 13 (ver capítulo 7, Figura 4).

Al abrir el compartimento estanco, las fuerza del aire en el vacío se lleva todos los objetos que, eventualmente, se hayan dejado en el suelo. Por un simple favor de los autores generalmente esto no sucede. Habría sido una faena hacer la aventura de forma que hubiera algún objeto esencial por el suelo en el momento de la apertura del compartimento estanco. A propósito, es obvio que no es posible sobrevivir en el vacío sin mono y casco.

Nos damos cuenta, al escribir estas líneas, de haber pasado por alto una posibilidad: si el aventurero deja en el suelo al segundo piloto antes de cerrar el compartimento estanco, éste no revive. En la respuesta a la frase "Aprieta el verde", hay que prever un "if lugar (Segundo) = O or lugar (Segundo) = LU...". Ninguno de los revisores del programa se ha quedado atrapado aquí, pero es mejor preverlo porque si no la solución de la aventura puede ser imposible.

El reactor

La sala de control del reactor es, probablemente, el lugar de la nave espacial donde es posible realizar el mayor número de acciones. Como ustedes saben, la acción final de la aventura consiste en desactivar el reactor en base a informaciones recabadas de un manual incompleto.

Naturalmente, habrán comprendido que para desactivar el reactor hay que ejecutar exactamente al revés la secuencia de puesta en marcha del mismo. Dado que para esto último hay que tirar de la palanca, apretar el pulsador verde, apretar el pulsador amarillo y, por último, apretar el pulsador rojo, la secuencia opuesta será: apretar el pulsador rojo, apretar el pulsador amarillo, apretar el pulsador verde y, obviamente, empujar la palanca.

Parece que los aventureros no demasiado expertos tienen dificultades para imaginar una palanca típica de cuadro de control del tipo de las usadas para manejar las grúas o las excavadoras.

Volveremos sobre el tema de la dificultad cuando hablemos de la revisión y la puesta a punto. Por ahora, nos interesa explicar el mecanismo de funcionamiento del cuadro de mandos:

- si una orden se ha dado en la secuencia correcta, incrementa un contador;
- si una orden se da fuera de la secuencia correcta, pone a cero el contador y reduce a la mitad el tiempo disponible (recalentamiento por maniobra errónea).

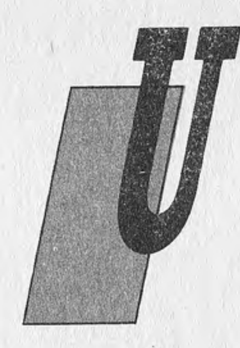
Del factor tiempo hablaremos en el capítulo próximo: por ahora les diremos que la variable T1 (timer-temporizador-1) es decrementada automáticamente en cada ciclo; cuando llegue a cero todo explotará silenciosamente (en el vacío). Dividiendo por 2 T1 se reduce a la mitad el tiempo disponible. El "div" en la expresión "timer 1 = timer 1"div" 2" significa división entera, o sea, sin resto (¿Cómo se hace en BASIC? Miren la línea 3830).

Volviendo al mecanismo de control del reactor, la variable V1 (var 1 en los apuntes) es el contador de desactivación, que al comienzo del juego vale cero. La acción "Aprieta Rojo" (la primera de la secuencia) pone V1 = 1 sólo si anteriormente valía cero, o sea, si no se había ejecutado ninguna otra maniobra (o si una maniobra equivocada había puesto todo a cero); la acción "Aprieta Amarillo" (la segunda) pone V1 = 2 sólo si valía 1, o sea, si se había apretado el rojo y sólo el rojo; "Aprieta Verde" pone V1 = 3 sólo si valía 2, o sea, después de la secuencia rojo-amarillo; finalmente, "Empuja Palanca" acaba el juego sólo si V1 valía 3, o sea, sólo después de la secuencia correcta rojo-amarillo-verde. Cada maniobra fuera de la secuencia envía a la línea 3830 del programa, que parte por la mitad el temporizador T1 y pone a cero V1, obligando a comenzar la secuencia desde el principio.

A propósito: el traje sirve solamente para impedir que un aventurero malintencionado consiga, al observar en el indicador de temperatura el efecto de las diferentes maniobras (no lo habían pensado ¿eh?), desactivar el reactor sin haber salvado primero al segundo piloto: la penúltima maniobra ("Aprieta Verde"), hace que el personaje muera por exceso de radiación (así aprende a no pasarse de listo).

CAPITULO XI

ÚLTIMOS DETALLES



Una vez definido el guión, preparen el diccionario, el plano, la lista de objetos, la tabla de las acciones y las subrutinas que las llevan a la práctica; estamos casi preparados para el "montaje" final de la aventura, en la forma de un programa en BASIC. Sólo faltan las subrutinas de introducción, de final de juego (para bien o para mal), y de paso del tiempo. Veámoslas.

Prolegómenos

La introducción se encuentra en el programa entre las líneas 1540-1670. Naturalmente, puede tener un número diferente la línea inicial, pero hay que acordarse entonces de modificar la llamada del intérprete en la línea 720. En primer lugar impriman los títulos y un mensaje introductorio a la aventura, después inicialicen las variables que el intérprete deba conocer; en nuestro caso, el lugar inicial y las variables especiales utilizadas por esta aventura (T1, V1 y V2, o sea, el timer, el contador del reactor y el estado del compartimento estanco).

Precisamente la línea 1670 hace este trabajo: LU = 6 significa que el aventurro empieza el juego en el lugar 6 (la cabina del comandante), T1 = 100 quiere decir que la aventura se resuelve en 100 turnos (salvo maniobras erróneas, que reducen el tiempo disponible). V1 = 0 y V2 = 0 son instrucciones superfluas, dado que en BASIC todas las variables numéricas contienen cero al princi-

pio del programa, pero no cuesta mucho indicarlo de manera explícita.

En la subrutina de "Save", hay que acordarse de grabar en disco o en casete no sólo LU y LO%, sino también estas variables: T1, V1 y V2. Lo mismo vale para la subrutina de "Load" (ver líneas 2000 y 2060, la sintaxis depende del ordenador utilizado).

Finales

El final positivo (aventura resuelta) no presenta problemas: basta con imprimir el mensaje de enhorabuena adecuado y terminar el programa con la instrucción END. Esto se puede hacer directamente en la subrutina que ejecuta la acción final (en nuestro caso, "Empuja Palanca"), como se puede ver en las líneas 3640-3770. El GOSUB 1120 de la línea 3650 sirve para evitar que una serie de mensajes demasiado larga desaparezca de la pantalla antes de que el jugador haya tenido tiempo de leerla. Úsenla todas las veces que lo crean oportuno.

El final negativo (muerto) también imprime un mensaje adecuado (líneas 1200-1320), después pregunta si se quiere volver a empezar desde el principio. En caso afirmativo, un simple "RUN" hace volver a empezar el programa, si no un "END" lo termina. Las líneas 1330-1360 son reutilizables tal y como aquí aparecen. Nótese que la 1340 extrae el primer carácter de la respuesta, de forma que acepte "S", "SI", "N", "NO"; no se aceptan otros caracteres iniciales, pues volvería a proponer la pregunta.

La subrutina de "muerto" sólo es llamada por otras subrutinas del programa (ejemplo: línea 3130), no por el intérprete. Por lo tanto, no hay ningún problema en cambiar los números de la línea, sólo hay que tenerlo en cuenta en las correspondientes llamadas. Además, dado que la subrutina acaba con "END" o con "RUN", se la puede llamar tranquilamente con un GOTO desde cualquier nivel de subrutina (no hay problemas de RETURN).

Temporización

Veamos finalmente el mecanismo de paso del tiempo, que encontrarán en las líneas 1400-1500. En caso de modificar los números de líneas, es necesario cambiar también la llamada a la línea 780 del parser.

La temporización está realizada a medida para esta aventura, pero el principio es válido para todos los casos en los que haya paso de tiempo (por ejemplo, una vela que se consume). El parser llama a la subrutina de tiempo cada vez que el jugador introduce una frase. La subrutina decrementa el tiempo restante (línea 1400) y después verifica si es el momento en que debe ocurrir alguno de los acontecimientos predispuestos, que en este caso son distintos mensajes a imprimir (líneas 1410-1430) y el mensaje final de destrucción cuando el tiempo ha vencido definitivamente. Aparte de este último caso, la línea 1440 normalmente vuelve al parser.

Montaje

Ahora que están todos los elementos, se puede proceder a la construcción del programa completo. Les sugerimos un procedimiento para el montaje de sus aventuras. Está claro que es conveniente usar de vez en cuando el "SAVE", si no les faltará seguramente la corriente, debido a la primera ley de Murphy ("Si algo puede ir mal, irá mal"). Por el mismo motivo no empleen siempre el mismo nombre, sino nombres y números correlativos (ejemplo: CAVERNAS1, CAVERNAS2, etc.). Si disponen de casete graben alternativamente en las dos caras. Cada cinco o seis grabaciones hagan una copia en otro disco o casete.

¿Están preparados? Pues adelante:

1) Carguen el intérprete en la memoria, o sea, las líneas desde la 100 a la 1160.
2) Introduzcan las líneas con los DATA del diccionario, comprobando una vez más que los vocablos estén en orden alfabético, seguidos cada uno de su propio código, y que no falte el "FD" al final. Les conviene utilizar un número de línea bastante alto, digamos 8000, para no obstaculizar después la escritura de las acciones.
3) Introducir las líneas con los DATA del plano, verificando que estén en el orden correcto (a partir del lugar 1), que cada descripción esté seguida de la lista de los pasos y, sobre todo, que esta lista sea correcta (es fácil equivocarse con secuencias de 12 cifras). No olviden "FP" al final. Utilicen números de línea sucesivos a los del diccionario, dejando algo de separación (digamos 300) para posibles (o sea, seguros) añadidos sucesivos.
4) Introduzcan las líneas con los DATA de las acciones (la tabla de las acciones), controlando que cada una comprenda número de síntesis y número de acción. Verifiquen muy cuidadosamente que los números de síntesis estén en orden correlativo, y no olviden "FA" al final. Los números de línea deben ser sucesivos a los del plano, con la correspondiente separación.

5) Introduzcan las líneas con los DATA de los objetos, verificando que estén en el orden deseado (que habrán escrito en algún sitio), que estén en grupos de tres (descripción, código de diccionario, lugar inicial) y que al final no falte "FO". Vale la misma explicación para los números de líneas.
6) A continuación del intérprete (por lo tanto antes de los DATA) escriban las subrutinas de muerto, de tiempo y de introducción, dejando espacio para modificaciones. Aunque dispongan en su ordenador de RENUMBER, les conviene utilizarlo lo menos posible, porque les hará perder las referencias a las que están acostumbrados y para las que prepararon las diferentes subrutinas. Usenlo al final para reordenar todo. (Si están entre los afortunados usuarios del Microsoft Basic 2.0 en el Macintosh, no tendrán el problema de los números de línea.) Sitúen las llamadas del parser en las líneas 720 y 780, en la subrutina de introducción y en la de tiempo, respectivamente. Si no utilizan el tiempo escriban una subrutina constituida por un simple RETURN (mejor no alterar el parser). No se olviden, en la introducción, de poner en LU el número del lugar inicial.
7) Escriban el conmutador de acciones sobre el modelo de las líneas 1710-1750. La primera línea (ON.A.GOTO...) indica los números de líneas de las primeras diez acciones, la segunda (ON A-10 GOTO) los de las diez sucesivas, y así sucesivamente. La línea 1750 es muy útil durante la puesta a punto del programa: si el número de la acción a ejecutar es mayor que el número de líneas previstas en el conmutador, imprime un mensaje del tipo "ACCION 25" y vuelve. Para nuestra comodidad, es mejor indicar los números de línea de las acciones con intervalos de 50, más o menos, de forma que quede un espacio suficiente entre uno y otro.
8) Escriban las subrutinas que ejecutan las acciones generales, que pueden copiar de las nuestras (acciones 1-7, líneas 1770-2120). Tienen que modificar "Coge" (2) y "Deja" (3), quitando los añadidos que se refieren a la nave espacial. También tienen que modificar "Save" (5) y "Load" (6) para su ordenador, si no es un Apple II; no olviden guardar y cargar también los posibles temporizadores y/o las variables especiales. Naturalmente, pueden añadir las otras subrutinas generales previstas por ustedes (ejemplo "Mueve XXX", "Abre XXX", etc.): cuantas más haya más "inteligente" parecerá la aventura al jugador. Importante: verifiquen, y si es necesario modifiquen, los números de líneas indicados en el conmutador. Cada uno de ellos tiene que corresponder a la primera línea de la subrutina correspondiente (y no a un REM). En nuestro programa, las acciones 8 y 9 no se han usado, con el fin de mantener reagrupadas las acciones generales, dejando espacio para eventuales añadidos. Ustedes organícense como quieran.
9) Escriban las subrutinas que ejecutan las acciones específicas de su aventura, recordando siempre que tienen que empezar en el número de línea indicado por el conmutador y en la posición correspondiente al número de la subrutina (el primer número indica dónde empieza la primera subrutina, etc.).
10) Comprueben una vez más que todas las llamadas del conmutador ON...GOTO vayan a la primera línea de la subrutina correspondiente, y que cada subrutina acabe con RETURN o con un GOTO a otra subrutina.

La aventura está terminada; no queda más que comprobarla.

Eliminando errores (Debug)

Seguramente creían que habíamos acabado, pero no estamos más que a mitad del trabajo (bueno, un poco más de la mitad). Quedan por hacer dos operaciones que requieren bastante tiempo, pero que son fundamentales para la buena marcha de un juego de aventuras: el debug y la evaluación.

El debug, como saben, consiste en la búsqueda y corrección de errores. En el caso de una aventura en BASIC, los hay de dos tipos: errores de programación y errores de juego. Los primeros son bien conocidos: se manifiestan con el mal afamado "SINTAX ERROR", con otros mensajes más extravagantes o crean efectos extraños, como los que puede causar la falta de un RETURN al final de una subrutina. A propósito, si después de haber impreso "Un poco de paciencia..." el programa se bloquea, señalando un error, puede ser que el número indicado en el mensaje de error no sea efectivamente el de la línea que lo contiene. De hecho, un error en los DATA puede ser señalado, según los casos, en la línea efectiva de DATA donde está el error, en una línea de DATA sucesiva, o en una línea de la subrutina de lectura (1040-1070). Motivo de más para que se comprueben bien los DATA. Otro error típico es la referencia a un elemento inexistente de una matriz. Si sus aventuras prevén más de 100 vocablos en el diccionario, 30 lugares, 150 acciones o 50 objetos, deberán modificar las líneas 970-1000 que establecen la dimensión máxima de cada matriz.

Dado que esto no es un curso de BASIC, los otros posibles errores del programa deberán encontrarlos ustedes mismos.

Para encontrar los errores de juego sólo hay un método: jugar la aventura probando dentro de los límites posibles, todas las combinaciones de sucesos y situaciones. Lo primero que hay que hacer es el control del plano: prueben elegir desde cada lugar todas las direcciones, incluidas las no permitidas, comprobando que coincidan con las previstas en su dibujo.

Cuando estén seguros de que el plano es correcto, comprueben el diccionario: la aventura tiene que reconocer todas las palabras que se han incluido. Esto sirve también para comprobar que han respetado el correcto orden alfabético en los DATA. Después verifiquen que los objetos se encuentran en los lugares donde deben estar, y comprueben que sólo se pueden coger los que deben ser cogidos. Al hacer esto también estarán comprobando la subrutina de "Coge".

Hemos llegado al punto más importante: la comprobación de las acciones. Tomen sus apuntes y, partiendo de la primera acción, prueben todas las posibles variantes de cada acción prevista. No olviden ninguna ("total, seguro que funciona"). Prueben absolutamente todo. En particular, comprueben a fondo la "secuencia principal", o sea, la cadena de acciones que lleva a la solución de la aventura. Una famosa casa de software estadounidense ha puesto en circulación una aventura no resoluble a causa de un error de este estilo (y ha tenido que sustituir varios discos): ocurre hasta en las mejores familias.

Para aligerar el trabajo de comprobación les servirá de ayuda lo siguiente: pueden parar el programa (con CTRL-C, STOP, BREAD o como se llame en su ordenador), cambiar el valor de las variables (especialmente LU, por ejemplo LU=5) y hacer CONT. De esta manera pueden cambiar el estado del juego sin tener que hacer secuencias de acciones complejas. Esto es particularmente cómodo en el caso de aventuras de dimensiones medias, donde se necesita mucho tiempo para conseguir un determinado objeto o llegar a cierto sitio. Usen también el "Save" para guardar la situación antes de probar todas las variantes en un determinado lugar (y de paso así comprueben también "Save" y "Load"). Una advertencia: si paran el programa durante el INPUT (línea 800), en algunos ordenadores no es posible continuar con CONT. Tienen que escribir, en cambio, GOTO 800 para continuar correctamente.

Cuando estén bien seguros de que todo funciona correctamente, estarán preparados para pasar a la segunda fase.

Evaluación

La evaluación es una tarea que ustedes mismos no pueden hacer. No estamos infravalorando su habilidad, sólo queremos decir que la valoración de una aventura (como de cualquier juego) no la puede hacer el autor. En efecto, la evaluación sirve para establecer cómo es acogido el juego por sus destinatarios: los jugadores. Muchos autores omiten este paso fundamental, y mandan a las tiendas juegos técnicamente perfectos pero aburridos o injugables (y no sólo aventuras).

Para la evaluación necesitarán a uno o más amigos, a ser posible que no sean programadores (o que no hayan visto sus trabajos). Siéntenlos, uno por uno, delante del teclado, hagan empezar el programa y observen cómo actúan. Regla fundamental: ¡no digan nada! No deben ayudarles a resolver el juego; en cambio, deben tomar apuntes y escribir todas las frases que no habían previsto pero que merecen una respuesta, las situaciones demasiado difíciles, las que son demasiado fáciles, los "Uf, que aburrimiento", etc. Descubrirán que ese problema no era tan fácil como parecía, que ese chiste no era tan divertido, que la línea obvia de comportamiento sólo era obvia para ustedes, y demás cosas por el estilo. Descubrirán también que unos resuelven el problema A en seguida y abandonan en el problema B. Otros hacen exactamente lo contrario. Si hay demasiados que abandonan en el problema C, es el caso de hacer algún ajuste.

Sobre todo, recuerden que la finalidad de una aventura es la diversión del jugador, y no la satisfacción de decir "Jamás la ha resuelto nadie (je, je, je...)". Por otra parte, superar obstáculos demasiado fáciles no da satisfacción a quien juega. La única manera para encontrar el equilibrio justo es hacer jugar al máximo posible de personas, y calibrar cuidadosamente la dificultad y las ayudas.

A título de ejemplo, he aquí las principales variantes que la evaluación ha demostrado necesarias para "La nave espacial condenada":

- Aceptar también "Lee indicador", no sólo "Mira indicador". El indicador no tiene efecto sobre las acciones, pero es fundamental para la comprensión de la trama y para el suspense: tiene que ser visto (entre otras cosas, por eso hay dos).
- Añadir un objeto "Escalerilla que baja" en el lugar 4 (final pasillo S) que, obviamente, no se puede coger. Muchos jugadores, en efecto, no tienen la costumbre de explorarlo todo y tardan en encontrar el reactor (que tiene que hacer comprender el peligro y dar pie al rompecabezas).

- La acción final "Empuja palanca" es demasiado difícil para la media de los jugadores principiantes. Algunos, además, se imaginan una palanca tipo freno de mano y dicen "Baja palanca" y otras frases por el estilo. Por otra parte, no hay que facilitar demasiado la solución para no quitar la satisfacción de haberla encontrado. Un compromiso posible puede consistir en prever la acción "Mira palanca" y la correspondiente respuesta: "Es una palanca típica de cuadro de mandos, que se puede mover en dos direcciones". Si el jugador no mira la palanca, peor para él. En las instrucciones está escrito que "Mira" es una operación fundamental. A propósito: si quieren publicar una aventura, escriban tres o cuatro páginas con instrucciones sencillas y claras, dirigidas a un lector que no haya visto en su vida un juego de este tipo.

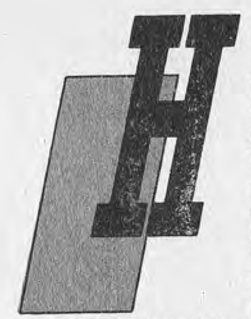
Al hacer añadidos y modificaciones al programa, acuérdense de respetar el orden alfabético en el diccionario y el numérico en las síntesis de las acciones. Los lugares y objetos nuevos, en cambio, pueden tranquilamente añadirse al final de los ya existentes (no en medio, porque si no cambian los números).

Finalmente

Ahora están en condiciones de escribir su propia aventura. Si no tienen algo claro, reléanse con calma el capítulo correspondiente, consulten las tablas contenidas en el apéndice y estudien el programa de "La nave espacial condenada". El capítulo siguiente describe el funcionamiento del intérprete, y el posterior ilustra posibles añadidos y mejoras para sus aventuras, por ejemplo gráficos y sonido.

CAPITULO XII

EL INTÉRPRETE



asta ahora habíamos dicho: utilicen el intérprete sin preocuparse de cómo funciona. Pero esperamos que sean lo suficientemente curiosos como para querer estudiar detalladamente su mecanismo. Es un programa en BASIC relativamente breve (unas 150 líneas, incluidos los comentarios), pero ligeramente sofisticado: algunas de sus características provienen del programa "Aventura en el castillo", otras provienen de nuestro estudio del lenguaje ADL-1 (ADventure Languaje 1). Es, o al menos querría serlo también, un ejemplo de buena programación: subrutinas breves, cada una de las cuales desempeña un cometido bien definido.

Si tienen un ordenador con un BASIC decente descubrirán que muchas subrutinas pueden ser simplificadas. Por otra parte, este programa se ha escrito para ser portátil, o sea, para funcionar también con los BASIC más pobres (como el del Commodore 64). En particular, la limitación de 80 caracteres por línea es bastante molesta.

Describiremos el programa partiendo desde el comienzo y siguiendo el flujo lógico, con regresiones de vez en cuando para describir las diferentes subrutinas. ¿Preparados? ¡RUN!

Para empezar

La línea 140 envía a la 710, o sea, al programa principal ("main program" o "main" a secas para los amigos). ¿Por qué el programa principal está al final? Por un motivo práctico: en la mayor parte

de los BASIC, el tiempo necesario para encontrar la línea indicada en una instrucción GOTO o GOSUB es tanto más larga cuanto más adelante esté la línea en el programa (la búsqueda se hace secuencialmente, partiendo desde el comienzo). Se deduce que para aumentar la velocidad de ejecución conviene poner las subrutinas más utilizadas al comienzo del programa. El programa principal, que no es llamado por nadie, puede quedarse tranquilamente al final o casi. En realidad, nuestras subrutinas no están dispuestas en el orden más conveniente, pero la claridad también tiene su importancia.

El main (línea 710) llama en primer lugar a la subrutina de inicialización o preparación de las variables, en la línea 970. Esta empieza por dimensionar las matrices utilizadas para contener el diccionario, el plano, la tabla de las acciones y los datos de los objetos (970-1000). Si una o más dimensiones son insuficientes para la aventura que quieren escribir, pueden aumentarlas tranquilamente (compatiblemente con la memoria disponible). También era posible leer en los DATA el número de elementos contenidos en cada matriz (ej.: READ FD), pero hemos preferido no hacerlo y especificar la dimensión con un número fijo (ej.: 100 para el diccionario). Esto es porque la mayor parte de los compiladores BASIC no aceptan variables en los DIM, sino que exigen un número (¿qué es un compilador BASIC?, ¿para qué sirve? Lo comentaremos en el próximo capítulo).

Las matrices numéricas son todas de tipo entero (como indica el símbolo % que sigue al nombre) para ahorrar espacio, excepto CA. En efecto, cada elemento de una matriz entera ocupa habitualmente 2 bytes de memoria, contra los 4-5 de una variable numérica normal. CA no puede ser de tipo entero, porque tiene que poder contener también números de seis cifras, y el contenido de una variable entera generalmente está limitado a 32767. Atención: no se ha dicho que haya ventajas, en términos de velocidad, con el uso de las variables enteras. Por ejemplo, en el Apple II y el C-64 las operaciones con variables enteras son, en realidad, más lentas que las otras (el valor es convertido a formato normal, se ejecuta la operación, y el resultado es reconvertido).

La línea 1010 establece el nombre de los archivos a utilizar para la grabación ("Save") de la situación actual. La 1020 no tiene ningún efecto, sencillamente recuerda que las cuatro variables indicadas valen cero al comienzo del programa (útil en caso de releerlo después de algunos meses). Las líneas 1040-1070 leen las informaciones de las líneas de DATA y las ponen en las matrices recién dimensionadas. Dado que son sustancialmente idénticas, describimos una: la 1040. En primer lugar, se lee un dato en la variable alfanumérica A$; si el dato es "FD", la lectura del diccionario está terminada y el programa puede continuar. De otra manera, o

sea, si el dato es distinto de "FD", la variable FD (número de datos en diccionario) es incrementada (hay un dato). El elemento número FD (número 1 en este caso) de la matriz DZ$, o sea, el primer vocablo, es el dato leído (A$), mientras que el dato sucesivo, su código, puede ser leído directamente desde los DATA (con un READ) en el correspondiente elemento de la matriz DZ. Habiendo leído así un vocablo y su código, el GOTO 1040 repite el ciclo desde el principio (acaba sólo cuando encuentra el final del diccionario, indicado por el dato "FD").

El mismo sistema es utilizado en las líneas 1050, 1060 y 1070, con una particularidad en la 1060: habiendo leído un dato alfanumérico (cadena) en A$, y habiendo aclarado que no es "FA", hay que poner en la matriz CA no una cadena, sino un número. Dado que en A$ se ha leído ya una cadena, ésta se transforma en número con la función VAL. La inicialización ha terminado: la 1080 vuelve al programa principal, y también volvemos nosotros.

La 720 llama a una subrutina (ya vista) escrita por el autor de la aventura, que imprime los títulos, establece el lugar inicial (LU), prepara posibles temporizadores y variables especiales, y vuelve. La 730 llama a la cómoda subrutina de la 1120 (Aprieta <espacio> para continuar), que sirve para evitar que las informaciones en la pantalla "desaparezcan" de la vista antes de que el jugador haya tenido tiempo de leerlas. La línea 1130, que espera a que se apriete el espaciador, depende de la versión de BASIC, como ya se ha visto en el segundo capítulo. Así como está, sirve para el Applle II y el Commodore.

El analizador sintáctico (parser)

Las líneas 760-930 constituyen el bucle de juego, ocupado en gran parte por el parser (o analizador sintáctico), que junto a sus subrutinas constituye la parte más compleja del programa (aquí reside la "inteligencia"). La figura 1 muestra el diagrama de flujo simplificado. Sigámoslo a grandes rasgos, antes de analizar sus detalles más interesantes.

En primer lugar, se imprime la descripción del lugar en el que se encuentra el aventurero (760) y de los objetos presentes (770), después se llama a la subrutina de paso del tiempo (780), ya vista en el capítulo 11. La línea 800 efectúa la entrada de la frase del jugador, y la 810 llama al analizador del léxico, que envía el código de la primera palabra (verbo). Previo control (820) de que se ha escrito efectivamente una palabra (y no sólo artículos o espacios), la 830 se enfada si la palabra acaba por "r" (probablemente sea un verbo, como "coger"), advirtiendo que debemos tutear al ordenador (p. ej.: "coge"). La 840 verifica que la palabra esté entre

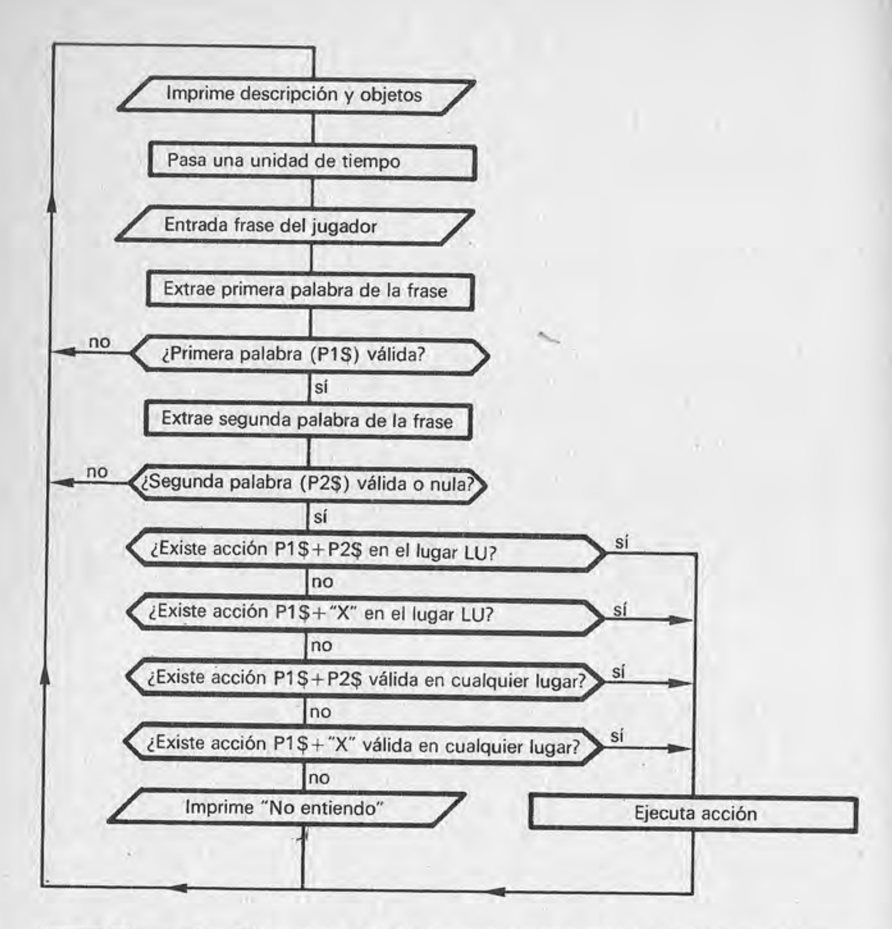


Figura 1.—Diagrama de flujo simplificado del analizador sintáctico.

las del diccionario, y la 850 llama de nuevo al analizador del léxico para conocer el código de la segunda palabra. También ésta se comprueba (860), recordando que puede ser nula (código cero) si la frase está formada por una sola palabra.

Si la segunda palabra es algo, puede ser un objeto: la 870 busca el posible índice en la lista de los objetos y lo pone en OB, para uso tanto del intérprete como del programador.

Ya está todo listo: no queda más que consultar la tabla de las acciones para ver si la acción indicada ha sido prevista. La línea 880 prepara los dos primeros tercios de la síntesis de acción, para

ejecutar las multiplicaciones una sola vez en vez de cuatro. Las líneas 890-920 buscan en la tabla los cuatro tipos posibles de frase prevista, en orden de precedencia (lean los REM). En el caso de una frase compuesta por una sola palabra, el código del nombre (C2) es cero y se ejecutan sólo la 890 y la 910 (las otras no tendrían sentido).

El test (IF A GOTO 760) al final de cada línea vuelve al principio del bucle en el caso de que la subrutina 500 haya reconocido una acción válida y la haya ejecutado en consecuencia (señalándolo con A diferente a cero). Si no se ha reconocido ninguna acción, la 930 imprime "No entiendo" y vuelve al principio del bucle.

Análisis del léxico

Veamos algunos detalles, comenzando por la subrutina de la línea 280, llamada en la 810 y en la 840. Analiza la frase IN$ introducida por el jugador (con una longitud de LI caracteres, ver línea 810) a partir del carácter número IN, extrae una palabra y consulta el diccionario para establecer el código correspondiente. Noten que el título de la subrutina (260) resume el trabajo: la palabra se restituye en P$ y el código en C.

La línea 290 se salta los posibles espacios iniciales, hasta llegar al principio de la palabra o al final de la frase (en este caso MID$ restituye una cadena vacía). Si en este punto el carácter a examen (el carácter número IN) se encuentra más allá del final de la frase no había ninguna palabra. En ese caso, la 300 prevé la restitución de una cadena nula en P$. Tratándose de una subrutina no demasiado sencilla, hemos preferido saltar al RETURN de la 360 en vez de poner un RETURN en la mitad de la subrutina, que sin duda habría causado problemas en caso de modificaciones sucesivas (dice el sabio: guárdense de las subrutinas con varios puntos de salida y, peor aún, de las que tienen varios puntos de entrada).

La línea 310 señala (en A) el punto de comienzo de la palabra, las líneas 320-330 la miden, carácter por carácter, hasta encontrar un separador, o sea, un espacio, o el final de la frase. En este punto, "A" indica el primer carácter de la palabra, e "IN" apunta al separador que acaba la palabra. La 340 pone la palabra en P$ y consulta el diccionario para determinar el código, la 350 descarta con desdén los artículos (código 7) y hace recomenzar la búsqueda de la palabra siguiente, como si el artículo no existiese.

La subrutina de búsqueda en el diccionario (180-240) no es más que una subrutina de búsqueda binaria: el diccionario es "abierto por la mitad" y, tras un vistazo, se descarta la mitad que

no puede contener la palabra (recordamos que están en orden alfabético); así se va partiendo por la mitad, cada vez, la parte que puede contenerla. En pocos pasos, o la palabra es encontrada y la línea 190 restituye el código leyéndolo en la matriz correspondiente DZ, o no está en el diccionario y la 230 restituye el código cero. "I" y "F" señalan el comienzo y el final de la zona del diccionario en la que ocurre la búsqueda. La línea 180 los coloca en los límites del diccionario, después la zona se estrecha cada vez más. Con 100 palabras, el vocablo es encontrado (o se sabe que no existe) en sólo siete ciclos de la subrutina.

Acciones y objetos

Cuando el analizador sintáctico ha preparado ya una síntesis de la frase y quiere saber si existe (y eventualmente ejecutar) la acción correspondiente, llama a la subrutina 500-550 (las llamadas están en las líneas 890-920). Esta subrutina busca en la tabla de las acciones, utilizando la subrutina 410-460 (que hace una búsqueda binaria idéntica a la del diccionario, ¿recuerdan que las acciones están ordenadas por número de síntesis?).

Si la acción no ha sido encontrada, la 500 vuelve con A=0, y el parser pasará a examinar la próxima posibilidad en orden de precedencia, o imprimirá "No entiendo" si ya las ha examinado todas. Si, en cambio, la acción existe en la tabla, A contiene el número. En este punto puede ser que sea necesario comprobar la presencia del objeto citado (OB): si el código de acción (A) es positivo, o bien si no hay una segunda palabra, la 510 se salta esta comprobación. En caso contrario (o sea, si hay una segunda palabra o el código de acción es negativo —la 500 muestra que no puede ser cero—), la 520 comprueba si el objeto está presente. Si OB vale cero no hay ningún objeto y la subrutina vuelve con A=1, es decir, como si se hubiera ejecutado correctamente una acción. En efecto, se ha impreso la frase "Aquí no está", y el parser no tiene que hacer posteriores búsquedas en la tabla.

La ansiada línea 530, que ejecuta la acción, se alcanza sólo en dos casos:

- la acción ha sido encontrada en la tabla y tiene código positivo;
- la acción ha sido encontrada en la tabla, tiene código negativo y la segunda palabra se refiere a un objeto presente.

En el segundo caso, la línea 520 ha cambiado de signo el código, que, por lo tanto, es positivo en cada caso. La 530 ejecuta la acción "A" llamando al conmutador de la línea 1710, y después vuelve indicando "misión cumplida" (A=1).

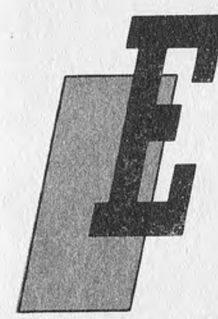
La subrutina 590-610 imprime la lista de los objetos presentes en el lugar L, cada uno precedido por la frase contenida en P$. Es utilizada por el main (770) para enumerar los objetos existentes al lado del aventurero, precedidos de "Veo" (ejemplo: "Veo un pulsador rojo"), y por la subrutina de inventario (2120) para enumerar los objetos presentes en el lugar cero, o sea, poseídos por el aventurero. La subrutina se limita a comprobar la lista de los objetos para verificar si el lugar donde se encuentra cada uno de ellos coincide con L. La función ABS sirve para ignorar el posible signo menos delante del número del lugar, que significa "que no se puede coger" (por ejemplo, la cama).

La breve pero importante subrutina 650-670, llamada por el parser en la 870, es la que pone en OB el número (índice en la matriz) del objeto citado como segunda palabra, pero sólo si éste está presente o transportado (si no, pone OB=0). Funciona de manera similar a la anterior, escrutando la lista de los objetos para ver si encuentra uno que responda a las condiciones requeridas: código en diccionario igual a C2 y lugar igual a LU o a cero. Sin esta subrutina sería imposible referirse a un objeto en la forma IF LO%(OB)...

CAPITULO XIII

VARIACIONES SOBRE UN MISMO TEMA

Compilar



Este último capítulo está dedicado a las posibles variaciones, añadidos y mejoras que se pueden hacer en un programa de aventuras. La primera y más sencilla operación consiste en compilar el programa BASIC, es decir, transformarlo en un programa equivalente, en código máquina o en otra forma más eficiente. Este trabajo lo hacen automáticamente ciertos programas llamados, precisamente, compiladores BASIC.

Son programas complicados que, ciertamente, requieren la utilización de una unidad de disco (disk drive). Para los ordenadores de mayor difusión existe, por lo menos, un compilador BASIC, y para algunos hay una amplia gama. Por ejemplo, para el IBM y el Olivetti pueden utilizar el BASCOMM, para el Apple II el mejor es el viejo TASC, y para el Commodore 64 está el pasable PETSPEED.

Las ventajas obtenidas de un compilador son esencialmente dos:

- la velocidad de ejecución aumenta notablemente (incluso más de 10 veces);
- el programa original (o "fuente") ya no es legible.

La primera ventaja empieza a notarse con aventuras de un cierto tamaño y, sobre todo, con muchos objetos. La segunda es quizá más importante: el programa no puede ser leído y modificado por cualquiera, y el trabajo queda protegido (no de la copia, obviamente). Además, todos los comentarios son eliminados por

el programa y no ocupan espacio en el resultado final. Como compensación, casi todos los compiladores aumentan considerablemente las dimensiones del programa: si tienen poca memoria disponible, seguramente tendrán problemas.

Considerando que los compiladores difieren profundamente uno de otro, les proporcionamos una regla empírica: para un programa de longitud media, las dimensiones del código compilado son más o menos iguales a las del programa original comentado ampliamente. Partiendo de un programa sin REM (o sea, muy mal escrito) las dimensiones aumentan, generalmente, desde un mínimo del 20-30% hasta el doble del original. Si no quieren o no pueden compilar el programa, por lo menos pueden usar una utilidad de "crunch" (compactación, compresión). Se trata de un programa que, como mínimo, quita todos los REM y une el máximo de líneas posibles en una sola línea BASIC. Así, el programa se hace mucho más pequeño, ligeramente más rápido en la ejecución y, sobre todo, de difícil lectura.

En ambos casos, compilación o compactación, comenten ampliamente la versión inicial del programa. Dado que los comentarios se quitarán no les cuesta casi nada y simplifica mucho el trabajo, especialmente si hay modificaciones.

Mejoras del intérprete

El programa puede ser mejorado de muchas maneras. Por ejemplo, para permitir guardar (en disco o en cinta) varias situaciones con nombres diferentes. Basta con modificar adecuadamente las subrutinas de "Save" y de "Load". Además, sería la ocasión para tener en cuenta los posibles errores durante la grabación y posterior lectura (por ejemplo: disco completo, disco protegido, disco deteriorado, etc.), evitando molestas paradas del programa. Una advertencia para quien utilice el Apple II con Prodos: no utilicen la orden STORE para guardar las variables. Las matrices de cadena no leídas por los DATA introducen periódicas y antipáticas pausas en el programa. Dejen las cosas como están.

Si se sienten programadores expertos, prueben a añadir un control del número máximo de objetos transportables. Parece fácil, pero no basta con contar los objetos cogidos con "Coge" y los dejados con "Deja". Hay que tener en cuenta todos los cambios de lugar, desapariciones, etc. Naturalmente, no querrán hacer este control cada vez que un objeto sea desplazado en una subrutina de acción (ejemplo: LO%(7)=13). Traten de inventar algo mejor.

Algunos jugadores prefieren escribir en primera persona (ejemplo: "Aprieto el botón"). Nosotros encontramos más simpática la segunda persona, tambien porque introduce un tanto de esquizofrenia (disociación mental) y consiguientes conflictos psíquicos entre "la mente" (el jugador) y "el brazo" (el personaje) que contribuyen a la diversión. Si de todas formas quieren la primera persona, no tienen más que cambiar los verbos en el diccionario. Cuidado con los reflexivos ("Me tiro"): hay que ignorar el "Me". También pueden ignorar la última letra de un verbo, pero tengan cuidado con los irregulares (ej.: "Haces", "Hago").

Finalmente, una posibilidad interesante: el efecto de algunas acciones puede depender también del azar (gracias a la función RND). Como siempre, se trata de un arma de doble filo: si la cosa no está bien estudiada, se obtienen comportamientos completamente idiotas. Si se usa con parsimonia, puede contribuir a la animación del juego.

Color y sonido

Si su ordenador puede escribir en color en la pantalla, tienen la ocasión de añadir un poco de variedad y animación a los mensajes impresos por el programa. Pueden utilizar un color para las descripciones, otro para los objetos, uno distinto para las entradas y otro para los mensajes, reservando imprevistos cambios de color o de fondo para situaciones especiales (por ej., fondo negro en el espacio). Es fácil modificar el programa para ese fin. Basta con insertar las instrucciones para el cambio del color de los caracteres (que generalmente son diferentes entre un ordenador y otro) en los puntos apropiados:

- para las descripciones, al comienzo de la línea 760;
- para los objetos, al comienzo de la línea 590;
- para las entradas, al comienzo de la línea 800.

¿Han visto la ventaja de desarrollar un determinado trabajo en un solo punto del programa? Las modificaciones son mucho más sencillas. Para los mensajes, pueden cambiar el color después de la 800 o ponerlo siempre en su sitio después de haberlo modificado en los tres casos anteriores. También pueden distinguir los mensajes del parser (tipo "No entiendo", "No lo tienes", etc.) de los del juego, cambiando el color al comienzo del conmutador de acciones (1710), pero pensamos que el resultado no es de los mejores, desde el punto de vista del juego.

Aparte del color, pueden introducir algún efecto sonoro en los puntos más significativos de la aventura (por ejemplo, un silbido del aire en el compartimento estanco o la sirena de alarma del reactor). No llenen la aventura de sonidos y ruidos: hacen mucho más efecto si llegan inesperadamente.

Dibujos y gráficos

Desde el punto de vista del programa, una aventura gráfica es idéntica a una de tipo texto, cómo la que está descrita en este libro. La única diferencia es que las descripciones de los lugares se han sustituido por dibujos, así como también (en las aventuras bien hechas) las descripciones de los objetos. En compensación, los mensajes de respuesta generalmente soy muy breves, teniendo que estar en la parte de la pantalla no ocupada por las figuras.

Si hay en circulación muchas aventuras de tipo texto mal escritas, de tipo gráfico mal realizadas, hay una verdadera invasión, especialmente para los ordenadores domésticos más difundidos. Muchos creen que algún dibujo (a menudo aproximado) basta para tapar una sustancial carencia de inteligencia del programa y de fantasía del autor. Son particularmente molestas las aventuras cuyos problemas se resuelven con una sola frase, escrita con las dos únicas palabras admitidas (no funciona ningún sinónimo). Ya hemos hablado de esto, pero tenemos la impresión de que muchos autores utilizan los gráficos como simple tapadera de trabajos de baja calidad. Moraleja: Si no se creen capaces de escribir una buena aventura gráfica, renuncien a los dibujos y busquen la manera de que los jugadores se diviertan. Si, en cambio, se animan, les damos algunas ideas sobre las cuales trabajar:

- en primer lugar, consideren que en casi todos los ordenadores la utilización de los gráficos quita memoria al BASIC. Ténganlo en cuenta adecuadamente;
- una manera poco comprometida, pero pasablemente eficaz, para insertar gráficos en sus aventuras consiste en hacer aparecer, de vez en cuando, imágenes "a toda pantalla" de las escenas más significativas (por ejemplo, la sala del reactor de la nave espacial). Sencillamente se trata de tomar la imagen del disco cuando sea necesaria; si no tienen disk drive, o éste es tan lento como el hambre, olvídense de él (si son expertos, pueden trabajar en lenguaje máquina y tener alguna imagen en los "agujeros escondidos" de la memoria). La cantidad de imágenes puede aumentarse si tienen a su disposición un programa de utilidad que les permita tener las imágenes en formato comprimido (es decir, utilizando menos memoria);
- un método más sofisticado, utilizado en gran parte de las mejores aventuras, consiste en utilizar un "Lenguaje gráfico", en el cual la figura está descrita como un conjunto de puntos, líneas, áreas, y así suvesivamente. Por ejemplo, un rectángulo es memorizado simplemente con las coordenadas de dos vértices opuestos. Este tipo de lenguaje, que permite tener muchas imágenes en poca memoria, puede estar ya disponible en el ordenador en forma más (Macintosh) o menos (IBM, MSX) evolucionada, puede adquirirse en disco (ejemplo: Graphics Magician para Apple II) o puede estar escrito por ustedes mismos si tienen la paciencia o los conocimientos para hacerlo. Se requiere, como mínimo, que el ordenador tenga las instrucciones para dibujar puntos y trazar líneas. Si no las tiene, es necesario proveerse de una extensión de BASIC adecuada, como el BASIC Simon para el C-64 (por lo menos se puede utilizar sin complicados y lentísimos POKE);
- en las aventuras gráficas el problema principal está constituido por los objetos: resulta antipático dibujar un lugar sin mostrar los objetos presentes, o afrontar el arduo camino de superponer objetos sobre el dibujo sin chapuzas o perspectivas absurdas, o limitarse a un dibujo de los objetos presentes fuera de la figura principal. Para el inventario, la figura grande puede ser sustituida por figuritas representativas de los objetos transportados.

Aquí nos paramos. Hemos arañado apenas la superficie del problema; necesitaríamos otro libro para tratarlo de manera adecuada (podríamos también escribirlo; antes o después...). En esencia, el camino de las aventuras gráficas puede dar grandes satisfacciones y grandes desilusiones. Si lo quieren recorrer, primero practiquen con aventuras clásicas de tipo texto. Recuerden que los gráficos no pueden salvar nunca a una aventura mediocre.

Conclusión

Aquí acaba la aventura. Hemos intentado pasarles una parte de nuestra experiencia, ahora es su turno. Por nuestra parte, todavía tenemos muchas cosas que contarles y esperamos tener pronto la ocasión de hacerlo.

APENDICE A

VARIABLES PRINCIPALES DEL INTERPRETE

Diccionario:

FD	Número de vocablos en el diccionario.
DZ$(1..FD)	Vocablos en orden alfabético.
DZ(1..FD)	Código de los vocablos correspondientes a DZ$.

Plano:

FP	Número de lugares en el plano.
DE$(1..FP)	Descripción de los lugares en orden de número.
DI$(1..FP)	Conexiones del lugar descrito en el correspondiente elemento de DE$ con los seis lugares adyacentes (N/S/E/O/A/B).

Acciones:

FA	Número de acciones en la tabla.
CA(1..FA)	Códigos (síntesis) de acción en orden numérico.
AZ%(1..FA)	Número de las acciones correspondientes a ejecutar; negativo si se requiere la presencia del objeto citado.

Objetos:

FO	Número de objetos.
OB$(1..FO)	Descripción de los objetos en orden del número de objetos.
CO%(1..FO)	Código en el diccionario del objeto descrito en el correspondiente elemento de OB$.
LO%(1..FO)	Lugar del objeto correspondiente: 1..FP lugar indicado, se puede coger.

-1..-FP	lugar indicado, no se puede coger.
O	transportado por el aventurero.
-99	"limbo" (fuera del juego).

Varias:

LU	Lugar actual del aventurero (variable).
OB	Número del objeto citado como segunda palabra; cero si no está presente o es transportado.
P1$,P2$	Primera y segunda palabras válidas de la frase.
C1,C2	Códigos en el diccionario de P1$ y P2$.

APENDICE B

SINTESIS DE ACCION Y PRIORIDADES DE BUSQUEDA

Síntesis de acción = número de 6 cifras (LLPPSS):

Código del lugar (2) + Código primera palabra (2) + Código segunda palabra (2).
Ejemplo 034709: la acción es ejecutada si la frase compuesta por dos palabras con códigos de diccionario 47 y 9 se escribe en el lugar 3.

- **Primeras dos cifras (código lugar):**

00	Acción válida en cualquier sitio.
01..FP	Acción válida sólo en el lugar indicado.

- **Cifras centrales (código primera palabra):**

01..FD	Código de la primera palabra en el diccionario.

- **Ultimas dos cifras (código segunda palabra):**

00	Segunda palabra nula, frase de una sola palabra.
01..FD	Código de la segunda palabra en el diccionario.
99	(X). Vale cualquir palabra siempre que sea válida, o sea, existente en el diccionario y no ignorada (ej.: artículos).

Prioridades de búsqueda de acciones con frases de una palabra:

LLPPOO	Palabra escrita en el lugar LL.
OOPPOO	Palabra escrita en un lugar cualquiera.

Prioridades de búsqueda de acciones con frases de dos palabras:

LLPPSS	Pareja de palabras pronunciadas en el lugar LL.

LLPP99	Primera palabra, seguida de una palabra válida cualquiera, pronunciada en el lugar LL.
OOPPSS	Pareja de palabras pronunciadas en un lugar cualquiera.
OOPP99	Primera palabra, seguida de cualquier palabra válida, pronunciada en cualquier lugar.





a realización y utilización de los llamados "juegos de aventuras" es, sin duda, uno de los aspectos más atrayentes de los ordenadores. Con este libro de la B.B.I. le ofrecemos simultáneamente la posibilidad de acceder a:

- *un juego de aventuras completo;*
- *la técnica para diseñar sus propias aventuras sin excesivo trabajo;*
- *nuevos conocimientos sobre el BASIC.*

Para lograrlo nos basamos en el gran juego "La nave espacial condenada". Tras presentárselo y ofrecer su listado se conduce al lector más allá del escenario y las candilejas, pasando sobre los PRINT y los READ, para explicarle y enseñarle los trucos más útiles y curiosos que se emplean en este género.

Es suficiente un mínimo conocimiento del BASIC para ser capaz de diseñar a partir de aquí sus propios juegos de aventuras, aprovechando el "esqueleto" universal que incluimos. Los más expertos encontrarán también una descripción completa del funcionamiento del programa, lo que les dará la oportunidad de construir juegos más complejos y sofisticados.

395 pts.
(incluido IVA)